Revisiting Multimodal Transformers for Tabular Data with Text Fields

Thomas Bonnier
Centrale Lille Alumni, France
thomas.bonnier@centraliens-lille.org

Abstract

Tabular data with text fields can be leveraged in applications such as financial risk assessment or medical diagnosis prediction. When employing multimodal approaches to make predictions based on these modalities, it is crucial to make the most appropriate modeling choices in terms of numerical feature encoding or fusion strategy. In this paper, we focus on multimodal classification tasks based on tabular datasets with text fields. We build on multimodal Transformers to propose the Tabular-Text Transformer (TTT), a tabular/text dual-stream Transformer network. This architecture includes a distance-to-quantile embedding scheme for numerical features and an overall attention module which concurrently considers self-attention and cross-modal attention. Further, we leverage the two well-informed modality streams to estimate whether a prediction is uncertain or not. To explain uncertainty in terms of feature values, we use a sampling-based approximation of Shapley values in a bimodal context, with two options for the value function. To show the efficacy and relevance of this approach, we compare it to six baselines and measure its ability to quantify and explain uncertainty against various methods. Our code is available at https://github.com/thomas-bonnier/TabularTextTransformer.

1 Introduction

Tabular datasets with text fields can be leveraged in various critical applications such as finance or healthcare. In financial risk assessment, numerical, categorical, and text data can be used by a classification model in order to assess companies' creditworthiness (Nguyen et al., 2021). In the medical field, clinical data and caregiver notes could be employed for diagnosis prediction. Transformers with an attention mechanism have become popular by achieving state-of-the-art performance in natural language processing (Vaswani et al., 2017; Devlin et al., 2019). In particular, multimodal Transformers have been used with various modalities such as audio, language, and vision (Tsai et al., 2019).

Motivation. Employing Transformers with multimodal tabular/text datasets requires adequate modeling choices. First, relevant encoding schemes should be used for numerical features. The traditional mapping methods used to construct embeddings for these features, e.g. linear functions (Gorishniy et al., 2021), are more difficult to interpret in high-dimensional settings. Numerical features are ordered and can have various distributions including extreme values. This should be expressed through the choice of a relevant embedding scheme. Regarding the attention mechanism, it may be more relevant to consider the whole surroundings of an item from the input sequence rather than exclusively considering the features from the same modality or from the other modality. In other words, the attention weights may be more informative by taking into account the whole context rather than considering partial contexts (e.g. cross-modal attention only). Lastly, when a multimodal model makes predictions based on unlabeled data, it is essential for subject matter experts to understand whether the predictions are trustworthy. If the uncertainty is significant for a given prediction, the source of uncertainty should be explained in terms of feature values. For instance, a data scientist could identify regions of the feature space where the model is uncertain because it is under-specified by the data. Or a doctor might want to know why a diagnosis prediction is uncertain and might then override the prediction thanks to expert knowledge.

Perimeter and contribution. The focus is on multimodal classification tasks based on tabular datasets with text fields in English. These datasets contain numerical and categorical features (together referred to as the *tabular* modality here) and fields with free-form text (i.e. the *text* modal-

Findings of the Association for Computational Linguistics ACL 2024, pages 1481–1500

ity). Numerical features have continuous scalar values while categorical features have discrete values. The latter could be a finite number of unquantifiable categories (e.g. country), boolean values (e.g. True/False flag), or ordinal values (e.g. rating from 1 to 5). Our mains contributions are fourfold:

(1) We propose the Tabular-Text Transformer (TTT), a tabular/text dual-stream Transformer network. This model exploits a specific embedding scheme for each numerical feature value, based on the distances to a set of quantiles of the feature distribution. TTT also employs an *overall attention* module which simultaneously considers self-attention and cross-modal attention.

(2) The two modality streams of TTT can then be leveraged in order to quantify uncertainty when the model makes predictions based on unseen data.

(3) To explain local predictive uncertainty in terms of feature values, we leverage a sampling-based approximation of Shapley values in a bimodal context, with two options for the value functions: a similarity measure and a supervised learning approach.

(4) To show the relevance of our approach, we (a) assess it on eight classification datasets against six baselines including pretrained models, (b) perform various ablation studies, and (c) measure its ability to quantify and explain uncertainty against various methods.

2 Related Work

Embeddings for tabular features. The quality of feature embeddings is key for the performance of models such as Transformers. The objective is to find a map from a scalar value or a category name to an embedding vector of higher dimension. To mimic word embeddings, a categorical feature can be embedded through a look-up table. Numerical features can naturally be encoded by using linear functions (Gorishniy et al., 2021) or piece-wise linear encoding (Gorishniy et al., 2022).

Multimodal machine learning. For a given task, a multimodal model leverages heterogeneous and connected modalities like text, image, and audio, as inputs. Such a model aims to learn representations of cross-modal interactions by fusing information across diverse modalities (Liang et al., 2022; Xu et al., 2023). With early fusion, cross-modal interactions happen at an early stage (Guo et al., 2020). For a Transformer with early concatenation of two modalities, this means that full pairwise attention will be computed at all layers. At the other end of the spectrum, late fusion of final representations make cross-modal interactions occur at a later stage (Gu and Budhkar, 2021). While vectors can be fused with a simple addition or concatenation, Zadeh et al. (2017) propose the Tensor Fusion Network to represent the inter-modality dynamics by using a Cartesian product of embeddings. The Multimodal Transformer (Tsai et al., 2019) is a multi-stream model that takes advantage of cross-modal attention in order to attend to interactions between multimodal sequences at all layers. In this approach, self-attention is computed separately.

Uncertainty quantification and explanation. There are two types of uncertainty: aleatoric and epistemic (Antoran et al., 2021). The first type is caused by noise in the data, generating for instance class overlap. The second one originates from the model's parameters being under-specified by the data. In that case, out-of-distribution (OOD) instances can increase uncertainty during inference. To quantify predictive uncertainty, methods such as conformal prediction (Vovk et al., 2005; Papadopoulos et al., 2002; Bonnier, 2023) can produce prediction sets based on a target coverage level. To detect model failure during inference, Corbière et al. (2019) propose a method which estimates the true class probability in image classification tasks. Speaking of explanation methods, Parcalabescu and Frank (2023) present MM-SHAP, a multimodality score based on Shapley values, which helps detect unimodal collapse. To overcome exponential time complexity, Štrumbelj and Kononenko (2010) suggest an efficient sampling-based approximation of Shapley values (Shapley, 2016) and compute the feature contributions to a classifier's prediction in unimodal contexts. Antoran et al. (2021) propose CLUE, a method based on counterfactuals, which identifies which features are responsible for uncertainty in probabilistic models. They focus on unimodal tabular or image datasets.

3 Tabular-Text Transformer

We consider a dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, where each input $x = (x_{text}, x_{cat}, x_{num}) = (x_{text}, x_{tab}) \in \mathbb{X}$ contains text fields, categorical, and numerical features. The true class is $y \in \mathbb{Y} = \{1, 2, \ldots, C\}$ for C-class classification tasks.

3.1 Novel Components

3.1.1 Distance-to-quantile embedding

For a random variable X with a cumulative distribution function F_X, its quantile with probability α is defined by $F_X^{-1}(\alpha) = \inf\{x : F_X(x) \geq \alpha\}$. The objective of this module is to transform a scalar and standardized feature value $v_{num} \in \mathbb{R}$ into an embedding vector $z_{num} \in \mathbb{R}^{1\times d}$, as a weighted sum of quantile embeddings: $z_{num} = \omega^T S$. Here, d is the embedding dimension. $S \in \mathbb{R}^{s\times d}$ denotes the stack of trainable embeddings of s empirical quantiles $q_0, .., q_{s-1}$ of the corresponding feature distribution $\{v_{num,i} : i \in \mathcal{I}\}$, where the training dataset is indexed by a set $\mathcal{I}$. The quantiles are for s evenly spaced cumulative probabilities $(0/(s-1), 1/(s-1), ..., (s-1)/(s-1))$ for integer $s > 1$: e.g. $(0., 0.25, 0.5, 0.75, 1.)$ for $s = 5$. Quantiles are representative of the feature distribution and are used as references for the embeddings. $\omega \in \mathbb{R}^{s\times 1}$ is a normalized vector of weights. The latter are computed with the distances between the feature value v_{num} and the quantiles: (1) If $\exists k \in \{0, ..., s-1\} : v_{num} = q_k$, then $\omega_k = 1$ while the other elements $\omega_{j,j\neq k} = 0$; (2) Otherwise, $\omega_j = \frac{1}{|v_{num}-q_j|}$ for $j \in \{0, ..., s-1\}$. The weights are then normalized by dividing each element by the sum of weights. The resulting embedding vector of v_{num} will be influenced by closer quantiles. With this approach, an outlier value will be impacted by the embeddings of extreme quantiles, thus the method takes into account exceptional values. Further, high feature values will be more influenced by high level quantiles than low feature values, thus reflecting order. This method can be used to generate embeddings for each numerical feature individually.

3.1.2 Overall attention

α and β are two modalities with respective sequences of embeddings $Z_\alpha \in \mathbb{R}^{t_\alpha\times d}$ and $Z_\beta \in \mathbb{R}^{t_\beta\times d}$. $Z_{\alpha\beta} = [Z_\alpha||Z_\beta] \in \mathbb{R}^{(t_\alpha+t_\beta)\times d}$ is the concatenation of the previous two sequences. d denotes the embedding dimension for both modalities and t is the sequence length. For a single-head overall attention module, we define the Query $Q_\alpha = Z_\alpha W^{Q_\alpha}$, Key $K_{\alpha\beta} = Z_{\alpha\beta} W^{K_{\alpha\beta}}$, and Value $V_{\alpha\beta} = Z_{\alpha\beta} W^{V_{\alpha\beta}}$, with the projection matrices $W^{Q_\alpha} \in \mathbb{R}^{d\times d_k}$, $W^{K_{\alpha\beta}} \in \mathbb{R}^{d\times d_k}$, and $W^{V_{\alpha\beta}} \in \mathbb{R}^{d\times d_v}$. Motivated by the original definition of attention (Vaswani et al., 2017), we compute the overall attention for the stream of modality α,

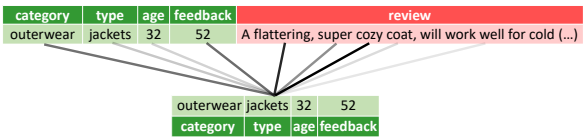

Figure 1: Illustration of overall attention between categorical feature *type* with value "jackets" and tabular/text elements, on *cloth* dataset (sentiment analysis).

in a bimodal setting:

$$\text{OverAtt}_\alpha(Z_\alpha, Z_{\alpha\beta}) = \text{Softmax}(\frac{Q_\alpha (K_{\alpha\beta})^T}{\sqrt{d_k}}) V_{\alpha\beta}$$

The output of the overall attention module is a sequence of embeddings of dimension $t_\alpha \times d_v$. Therefore, the modality α is enriched with its own features and the features from another modality β by concurrently considering self-attention and cross-modal attention. Figure 1 is a simple example of overall attention. It shows that a feature from the tabular modality (*type*="jackets") can attend not only to other tabular features such as the *feedback* count, but also to elements from the text modality such as the words *flattering* or *cozy*. Thus, the embedding vector of *type*="jackets" will be significantly impacted by the embeddings of those items, which may be more insightful than a limited context.

3.1.3 Uncertainty quantification

True labels are sometimes obtained with a certain cost or lag. Thus, it is key to measure the uncertainty when the model makes predictions based on new data. To optimize the parameters of TTT for a given classification task, we minimize the dual loss: $\mathcal{L}(\hat{l}_{text}(x), y) + \mathcal{L}(\hat{l}_{tab}(x), y)$, where $\hat{l}_{text}(x)$ and $\hat{l}_{tab}(x)$ are the logits predicted by the text and tabular streams of the model based on input x. $\mathcal{L}$ is the cross-entropy loss $\mathcal{L}(l, y) = -\sum_{c=1}^{C} \log(\text{Softmax}(l)_c)\delta_{cy}$ where l is a vector of logits, δ is the Kronecker delta, and $c \in \mathbb{Y}$. Our goal is to obtain optimized logits for each modality stream and employ them for uncertainty quantification. During inference, we can generate prediction sets based on the predictions from each modality stream. If each stream predicts a different label, and thus they disagree $(\arg\max_{c\in\mathbb{Y}}(\hat{l}_{text,c}(x)) \neq \arg\max_{c\in\mathbb{Y}}(\hat{l}_{tab,c}(x)))$, the decision will be uncertain. In that case, the prediction set will contain two predictions. Otherwise, the set will have a unique label. Prediction sets can be useful for subject matter experts who

need to obtain prediction regions rather than one single predicted label for any new input. Most importantly, this technique can be used to monitor uncertain predictions over time, by computing the mean prediction set size for a collection of new inputs and controlling its evolution.

3.1.4 Explaining uncertainty

Our objective is to understand why a given prediction is estimated as uncertain (i.e. the streams disagree) by comparison to a subset of predictions that are estimated as certain (i.e. the streams agree). In other words, we want to display the feature values that contribute to an uncertain outcome. This could be due, for instance, to an OOD input. Let $\hat{p}_\alpha(x) = \text{Softmax}(\hat{l}_\alpha(x))$ denote the softmax probability vector (i.e. vector of probability estimates) produced by the stream of modality α, with $\hat{l}_\alpha(x)$ the vector of logits predicted based on input x.

Constructing the value function. To compute the feature contributions to predictive uncertainty, we first need to define relevant value functions. Due to its binary nature, the stream disagreement indicator is not a tractable value function when estimating uncertainty. Therefore, we suggest two options for the value function in order to assess predictive uncertainty on a more granular scale.

When each stream predicts a different class, their respective softmax probability distribution should be more distant than when they agree. We thus evaluate the degree of uncertainty with the Jensen-Shannon Distance (JSD) (Lin, 1991), defined as: $JSD(p,q) = \sqrt{\frac{KL(p,m)+KL(q,m)}{2}}$, with here $p = \hat{p}_{text}(x)$ and $q = \hat{p}_{tab}(x)$ for input x. $m = \frac{p+q}{2}$ is a mixture distribution, and KL is the Kullback-Leibler divergence (Kullback, 1997). JSD is a symmetrized version of Kullback-Leibler divergence, bounded by 1 if KL uses base 2 logarithm. When the streams disagree, we expect that this metric will be higher than when they agree.

For the second value function, we fit a classifier $\hat{f}$ to the test dataset D_{test} indexed by $\mathcal{I}_{test}$, in order to predict the probability for a prediction to be uncertain, given the softmax probability distributions. This model is constructed as $\hat{f} = \mathcal{C}(\{((\hat{p}_{text}(x_i), \hat{p}_{tab}(x_i)), u_i) : i \in \mathcal{I}_{test}\})$, where $\mathcal{C}$ denotes any classification algorithm that takes in data indexed by $\mathcal{I}_{test}$ in order to output a classifier fitted on that data. Therefore, the concatenated probability estimates are used as features. The label u is equal to 0 (stream agreement) or 1 (stream disagreement). This classifier should be able to easily identify the degree of uncertainty based on the softmax probability distributions.

Approximating Shapley values for j-th feature. We build on the sampling-based method proposed by (Štrumbelj and Kononenko, 2010) to estimate the Shapley values by randomly and repeatedly selecting a subset of features instead of all possible coalitions. Based on the central limit theorem, the estimator is unbiased and its standard deviation is proportional to $\frac{1}{\sqrt{M}}$, where M is the total number of iterations. We adapt the method to the explanation of uncertainty in a bimodal context. The unlabeled test dataset is first split into certain (D_c) and uncertain (D_u) predictions based on stream agreement or disagreement, respectively. Let $x = (x_{text}, x_{tab})$ denote an input from D_u, where x_{text} is a sequence of token values and x_{tab} a sequence of tabular feature values. For x and for the j-th feature, we approximate the Shapley value to explain the uncertainty by performing M Monte Carlo iterations. Therefore, we compute the average contribution of a tabular feature with index j or a text feature (i.e. token) with index (i.e. position) j to the uncertainty, by iterating M times:

- We draw a random instance $z = (z_{text}, z_{tab})$ from D_c, where z_{text} is here a sequence of token values and z_{tab} a sequence of tabular feature values. D_c is used as the reference dataset to sample from as we want to understand what distinguishes x ($\in D_u$) from instances from D_c. For instance, if we consider the second value function, our objective is to uncover what contributes to $\hat{f}((\hat{p}_{text}(x), \hat{p}_{tab}(x))) - \mathbb{E}_{D_c}[\hat{f}((\hat{p}_{text}(X), \hat{p}_{tab}(X)))]$.

- We select a random subset of tabular feature indices s_{tab} and a random subset of token indices s_{text}. If the j-th feature is tabular, we also specify that $j \notin s_{tab}$; otherwise (j is a token index, i.e. a position in the text sequence): $j \notin s_{text}$.

- We construct two new instances. First, x_{+j} is input x where all the tabular feature values in x_{tab} with index in s_{tab} are replaced by the corresponding values from z_{tab}, and all the token values in x_{text} with index in s_{text} are replaced by the value of the padding token [PAD] when these token values are not in z_{text} (otherwise they are kept unchanged). As [PAD] does not contribute to the gradient, it is used as a mask.

Second, x_{-j} is similar to x_{+j} with one difference: if j is the index of a tabular feature, the value in x_{tab} with index j is also replaced by the corresponding value from z_{tab}; otherwise (i.e. j is a token index), the token value in x_{text} with index j is replaced by the value of the padding token [PAD] if this token value is not in z_{text}.

- We compute the marginal contribution for the given feature with each value function. For JSD, we compute the contribution for iteration m:

  $$\phi_j^m(x) = JSD(\hat{p}^+_{text}, \hat{p}^+_{tab}) - JSD(\hat{p}^-_{text}, \hat{p}^-_{tab}).$$

  With the the uncertainty classifier, we assess:

  $$\psi_j^m(x) = \hat{f}((\hat{p}^+_{text}, \hat{p}^+_{tab})) - \hat{f}((\hat{p}^-_{text}, \hat{p}^-_{tab})),$$

  Where $\hat{p}^+_\alpha = \hat{p}_\alpha(x_{+j})$ and $\hat{p}^-_\alpha = \hat{p}_\alpha(x_{-j})$ for modality α.

Lastly, we compute the average to obtain the approximated Shapley value for each value function: $\phi_j(x) = \frac{1}{M}\sum_{m=1}^{M}\phi_j^m(x)$ and $\psi_j(x) = \frac{1}{M}\sum_{m=1}^{M}\psi_j^m(x)$.

Disagreement between explanation methods. As the explanations provided by different methods may disagree (Krishna et al., 2022), it is essential to include various approaches in order to check the consistency between their outputs. In fact, our sampling-based algorithm makes it easy to compute the Kernel SHAP contributions (Lundberg and Lee, 2017). The "perturbation samples" x_{+j} and x_{-j} are converted into $z' \in \{0,1\}^T$, where T is the number of tabular and text features, and $z' = 1$ when the feature value from original x is present, and 0 when it is absent. For each z', we also have the corresponding JSD and uncertainty classifier values that have been previously computed with the sampling-based method. We compute the Kernel SHAP weights $\frac{(T-1)}{\binom{T}{|z'|}|z'|(T-|z'|)}$, where $|z'|$ is the number of present features with $0 < |z'| < T$. We fit a weighted Lasso regression $\hat{r}_1(z') = \phi_0^{ks} + \sum_{j=1}^{T}\phi_j^{ks} z'_j$ where the target values are the corresponding JSD values. We fit another weighted Lasso regression where the target values are the corresponding uncertainty classifier values: $\hat{r}_2(z') = \psi_0^{ks} + \sum_{j=1}^{T}\psi_j^{ks} z'_j$. Lastly, the coefficients ϕ_j^{ks} and ψ_j^{ks} are the resulting Kernel SHAP feature contributions computed for each value function. They can be compared to the ϕ_j and ψ_j previously computed with the sampling-based algorithm.

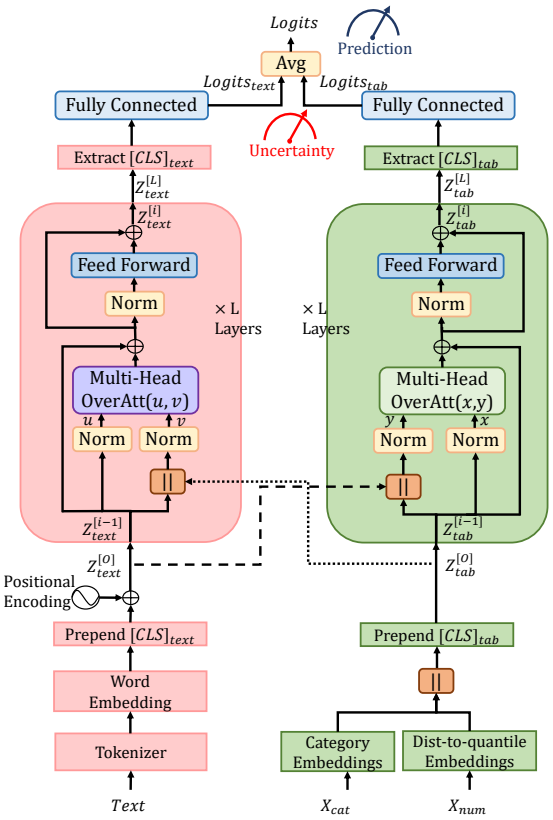

Figure 2: Overall architecture for TTT: dual-stream Transformer with distance-to-quantile embeddings, overall attention, and quantification of predictive uncertainty.

3.2 TTT Overall Architecture

To show how its components are connected, TTT's architecture is displayed in Figure 2.

Embeddings. The text inputs are transformed into word embeddings (dimension $\mathbb{R}^{t_{text}\times d}$) while categorical features are turned into category embeddings ($\mathbb{R}^{t_{cat}\times d}$). t_{text} and t_{cat} denote the number of text tokens and categorical features, respectively. These transformations are based on the usual mechanism of look-up tables. The embeddings ($\mathbb{R}^{t_{num}\times d}$) for the t_{num} numerical features are constructed by using the distance-to-quantile technique previously described. Categorical and numerical embeddings are concatenated ($\mathbb{R}^{t_{tab}\times d}$, with $t_{tab} = t_{cat} + t_{num}$). A classification token [CLS] (Devlin et al., 2019) is then added to the beginning of each text and tabular embedding sequence. All these embeddings are learned by the model through the training process. Lastly, fixed positional encoding with sine and cosine waves is added to the text embeddings as suggested by Vaswani et al. (2017). The inputs to the next step are denoted

$Z_{text}^{[0]} \in \mathbb{R}^{(t_{text}+1)\times d}$ and $Z_{tab}^{[0]} \in \mathbb{R}^{(t_{tab}+1)\times d}$ for the text and tabular streams.

Transformers with overall attention. Each Transformer contains L layers of overall attention blocks. If we consider for instance the tabular stream, each attention block with residual connection starts with the following operation for $i = 1, ..., L$ layers:

$$\text{OverAtt}_{tab}^{[i],mult}(\text{LN}(Z_{tab}^{[i-1]}), \text{LN}([Z_{tab}^{[i-1]}||Z_{text}^{[0]}])) + Z_{tab}^{[i-1]}$$

LN stands for layer normalization (Ba et al., 2016). $\text{OverAtt}_{tab}^{[i],mult}$ denotes the multi-head version (Vaswani et al., 2017) of OverAtt_{tab} at layer i. $Z_{text}^{[0]}$ are used as low-level features from the text modality. After applying layer normalization, the outputs are then fed to a feed-forward neural network. Lastly, we employ a residual connection.

Classification layers. The text and tabular embeddings of the [CLS] tokens ($\mathbb{R}^d$) from each output $Z_{\cdot}^{[L]} \in \mathbb{R}^{(t_{\cdot}+1)\times d}$ are projected through fully-connected layers to produce the logits for each stream. These are leveraged to quantify uncertainty as previously described. The final prediction is based on the argmax of the average of the text and tabular logits. The performance measurement is based on this final output. For the sake of clarity, the prediction set or the stream predictions denote the output of the uncertainty analysis while the (final) prediction denotes the single label predicted by the model in order to measure its accuracy.

4 Experiments

We empirically test the relevance of our approach on eight classification datasets. In the appendix, we provide further details on the experimental settings and results (e.g. details on datasets, data preprocessing, hyper-parameters, baselines).

4.1 Settings

Datasets. We use various public datasets for multi-class classification tasks, with a number of classes ranging from 5 to 100: *airbnb, cloth, petfinder, salary,* and *wine* with the 10/100 most frequent classes (referred to as *wine10* and *wine100*, respectively). We also test our method on two public binary classification datasets: *jigsaw* and *kick*. These datasets have been suggested by (Shi et al., 2021) and (Gu and Budhkar, 2021).

Model selection. Each dataset is split into training-validation-test disjoint subsets. The validation subset is used for hyper-parameter tuning and early stopping. For all non pretrained models, we select the best trial via a Bayesian optimization algorithm (Tree-structured Parzen Estimator) by using Optuna library (Akiba et al., 2019). We finetune the pretrained models based on DistilBERT-base-uncased (Sanh et al., 2019). The test dataset (20%) is used for the final evaluation (accuracy, uncertainty). Each use case is run over five different random dataset splits.

Evaluation 1 (low-dimensional embeddings). To reduce computation costs, we first evaluate various multimodal architectures (including TTT) with low-dimensional embeddings (maximum of 64): (1) EarlyConcat: A self-attention Transformer with early concatenation of modalities; (2) LateFuse: A Transformer with late fusion of modalities (Gu and Budhkar, 2021); (3) MulT: The Multimodal Transformer (Tsai et al., 2019) with cross-modal attention; (4) TFN: Tensor Fusion Network (Zadeh et al., 2017) with LSTM. For numerical features, TTT leverages the distance-to-quantile embedding scheme with an arbitrary value of 6 quantiles $(0., 0.2, 0.4, 0.6, 0.8, 1.)$. The embeddings of the baselines are constructed with linear functions (EarlyConcat, MulT) or features are kept as standardized values (LateFuse, TFN).

Evaluation 2 (pretrained models). Secondly, we test TTT-SRP: TTT with sparse random projections (Li et al., 2006) of input embeddings extracted from a pretrained model. TTT-SRP has an intermediate size for the embeddings, with a 240 dimensional space at most. It is initialized with pretrained DistilBERT-base-uncased input embeddings after reducing their dimensions through sparse random projections. We measure it against two pretrained baselines: (1) AllTextBERT: The tabular features, converted to strings, and the text fields are concatenated and input into DistilBERT-base-uncased as text; (2) LateFuseBERT: A dual-stream model with late fusion of [CLS] tokens generated by DistilBERT-base-uncased and a tabular Transformer.

Ablation studies. To evidence the contribution of each component, we perform three ablation studies for TTT: (1) Ablation 1: The numerical features are encoded with linear functions instead of the distance-to-quantile embedding scheme; (2) Abla-

Model	Model size	airbnb	cloth	jigsaw	kick	petfinder	salary	wine10	wine100
EarlyConcat	1.2M	0.326	0.635	0.885	**0.874**	0.372	0.450	0.800	0.652
LateFuse	1.2M	0.336	0.644	**0.891**	0.812	0.367	0.452	0.795	0.657
MulT	1.3M	0.363	0.636	0.888	0.872	0.376	0.454	0.803	0.653
TFN	1.3M	0.357	0.601	0.801	0.864	0.368	0.458	0.714	0.558
TTT (ours)	1.3M	**0.383**	**0.655**	**0.891**	0.871	**0.389**	**0.472**	**0.817**	**0.671**

Table 1: Evaluation 1. Accuracy on the test dataset averaged over 5 random seeds. Top results are in **bold** (higher is better). The variability in results is displayed in Appendix F. Average model size is given in million parameters.

Model	Model size	airbnb	cloth	jigsaw	kick	petfinder	salary	wine10	wine100
AllTextBERT	66.4M	0.309	**0.680**	**0.906**	0.868	0.346	0.440	**0.833**	**0.704**
LateFuseBERT	81.3M	0.320	0.670	0.904	0.853	0.359	0.429	0.817	0.683
TTT-SRP (ours)	8.2M	**0.383**	0.663	0.892	**0.874**	**0.379**	**0.471**	0.827	0.689

Table 2: Evaluation 2. Accuracy on the test dataset averaged over 5 random seeds. Top results are in **bold** (higher is better). The variability in results is displayed in Appendix F.

tion 2: We use self-attention instead of overall attention; (3) Ablation 3: We minimize the loss based on the average of logits: $\mathcal{L}((\hat{l}_{text}(x) + \hat{l}_{tab}(x))/2, y)$, instead of the dual loss. Moreover, to show the effect of the TTT-SRP's initialization scheme, we measure its performance against TTT-PCA (same as TTT-SRP but with principal component analysis instead of sparse random projections) and a standard approach based on the Kaiming uniform weight initialization (He et al., 2015).

Quantifying and explaining the uncertainty. Our method, used to quantify uncertainty (stream disagreement), is compared to the LAC conformal prediction approach (Sadinle et al., 2019) and the Monte Carlo dropout (MCD) with 50 simulations (Gal and Ghahramani, 2016). For LAC, we compute the mean prediction set size. For MCD, the uncertainty is assessed with the entropy. Lastly, we explain the uncertainty of TTT model's predictions from *wine100* test dataset. The dataset task is to predict grape varieties based on numerical features (e.g. price), categorical variables (e.g. country), and wine tasting descriptions. The uncertainty classifier is a random forest algorithm (Breiman, 2001) with 100 trees. In the sampling-based algorithm, we stop the sampling iterations when the maximum absolute change in the Shapley values computed with the stream disagreement (binary) signal is lower than 0.01. This method is used to speed up the computation as additional iterations would modify the Shapley values only very marginally. We use the binary signal for this purpose as the changes are expected to be more sudden than for the other value functions. Thus, the convergence is more certain when it occurs with the binary signal.

Dataset	Disagr. Acc.	Set size (SD)	Set size (LAC)	Entropy (MCD)
airbnb	0.32	1.59	1.49	2.01
cloth	0.40	1.19	1.18	0.98
jigsaw	0.54	1.03	1.03	0.34
kick	0.54	1.05	1.04	0.38
petfinder	0.32	1.43	1.38	1.74
salary	0.38	1.35	1.28	1.61
wine10	0.46	1.14	1.11	0.70
wine100	0.36	1.26	1.18	1.40

Table 3: TTT's accuracy when the streams disagree (Disagr. Acc.). Mean prediction set size for TTT (SD: Stream Disagreement, ours) and the conformal prediction baseline (LAC). Mean entropy for the Monte Carlo dropout baseline (MCD). All results are averaged over 5 seeds on the test dataset.

4.2 Results

Performance. The results from Table 1 show that TTT outperforms the other multimodal approaches on most datasets. Further, the results from Table 2 demonstrate that the lightweight TTT-SRP (8.2M parameters) outperforms LateFuseBERT (81.3M). TTT-SRP also matches AllTextBERT (66.4M) in terms of number of top results. Lastly, it is worth noting that TTT achieves better results than larger architectures from Table 2 on small datasets (i.e. *airbnb, petfinder, salary*).

Uncertainty quantification. Tables 1 and 3 show that the mean prediction set size is negatively correlated with TTT's accuracy. The Pearson correlation between the mean prediction set size and model's performance (across the 40 experiments: 8 datasets $\times$ 5 seeds) is slightly more negative (-

Model	airbnb	cloth	jigsaw	kick	petfinder	salary	wine10	wine100
TTT	**0.383**	0.655	**0.891**	**0.871**	0.389	**0.472**	**0.817**	**0.671**
Ablation 1	0.355	**0.658**	**0.892**	**0.870**	0.376	0.463	0.815	**0.672**
Ablation 2	0.357	0.641	0.882	0.862	0.379	0.456	0.798	0.640
Ablation 3	0.350	0.648	0.887	**0.870**	**0.398**	0.458	0.803	0.655

Table 4: Ablation studies. Accuracy on the test dataset averaged over 5 random seeds. Top results are in **bold**.

Model	airbnb	cloth	jigsaw	kick	petfinder	salary	wine10	wine100
TTT-SRP	**0.383**	**0.663**	0.892	**0.874**	**0.379**	**0.471**	**0.827**	0.689
TTT-PCA	0.368	0.659	**0.894**	**0.873**	0.369	0.456	**0.827**	**0.690**
TTT-Kaiming	0.377	0.657	0.889	**0.873**	**0.379**	0.453	**0.826**	**0.691**

Table 5: Accuracy with different initialization methods for the input embeddings, on the test dataset computed over 5 random seeds.

0.953) with our approach than with the other baselines: conformal prediction (-0.947) and Monte Carlo dropout (-0.932). The mean prediction set size computed with our technique is most relevant to quantify uncertainty on unlabeled data. Thus, when the streams disagree in their outputs, the final prediction is less reliable, and this leads to lower accuracy (first column of Table 3).

Ablation studies. Table 4 shows that TTT has a solid performance as it ranks first or second across the various datasets. The effect of using overall attention instead of only self-attention is quite obvious (Ablation 2). The dual loss used in TTT is also relevant based on the weaker results of Ablation 3. The effect of the distance-to-quantile encoding is less obvious for some of the datasets but may remain of relevance for datasets with many numerical features, such as *airbnb*. Lastly, Table 5 displays the results related to the initialization scheme of input embeddings: TTT-SRP achieves the best accuracy on most datasets.

Explaining the uncertainty. This method can be used on unlabeled datasets. Here, the true labels are only leveraged to audit the method. Figure 3 (left) displays the distributions of the JSD metric and uncertainty classifier's predicted probability over D_c and D_u, respectively. The uncertainty classifier performs better at discriminating between

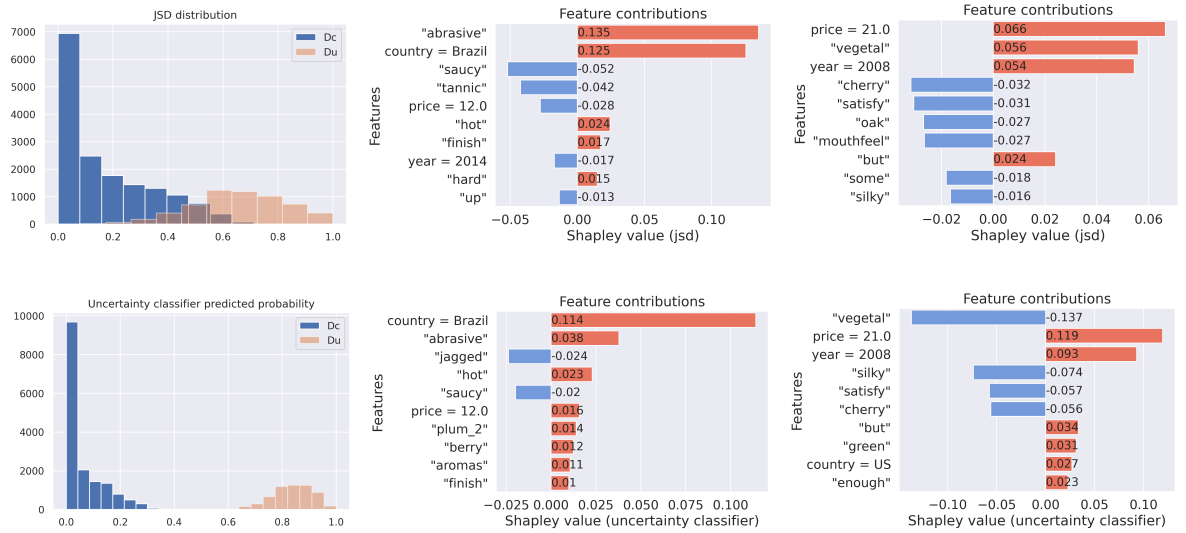

Figure 3: The first and second rows correspond to the JSD and uncertainty classifier, respectively. *Left*: Distributions of value functions' outputs by subset D_c and D_u. *Middle*: The top 10 feature contributions to uncertainty for an uncertain prediction due to data scarcity and potential class overlap. Positive contributions are displayed in red. *Right*: The top 10 feature contributions to uncertainty for an input where the *price* has been synthetically halved so that the prediction becomes uncertain. Unlike the tabular stream, the text stream still predicts a correct label.

certain and uncertain predictions, and may be more reliable. Figure 3 (middle) displays the results for an uncertain and incorrect final prediction. Each stream predicts a different label. The instance is related to a *country* value ("Brazil") which is quite rare in the dataset (i.e. data scarcity), generating an uncertain decision. Further, the term *abrasive* also contributes to the uncertainty. This word appears in text fields related to several grape varieties from the same region (e.g. *Merlot* or *Tannat* varieties). This is a sign of potential class overlap. Both value functions help detect these problems. To confirm the consistency of explanations, we compute the Pearson correlations between the Shapley values obtained with our method and with Kernel SHAP, for both value functions: 0.93 (JSD) and 0.86 (uncertainty classifier). Lastly, we perform a synthetic sensitivity test with an input where the *price* has been artificially halved so that the decision becomes uncertain (stream disagreement), but remains correct. Figure 3 (right) evidences the origin of uncertainty and the potential inconsistency between the *price* and *year* values. With the uncertainty classifier, these positive contributions are largely offset by the negative contributions of specific words (e.g. *vegetal*). This is why, unlike the tabular stream, the text stream still predicts a correct label. In that case, using the uncertainty classifier produces more reliable outputs as confirmed by the Pearson correlations between the feature contributions generated by our method and by Kernel SHAP: 0.39 (JSD) and 0.88 (uncertainty classifier). This explanation method could be employed on more critical applications such as medical diagnosis prediction by combining a predictive algorithm with human-in-the-loop. For instance, a prediction could be uncertain due to specific words from the caregiver notes. This would help a medical expert understand why the prediction is not certain and make a decision for a more likely diagnosis. Appendix K displays additional examples from *wine*, *cloth*, and *kick* datasets.

5 Conclusion

We presented the Tabular-Text Transformer, a multimodal approach which shows promising results on classification tasks. We demonstrated how uncertainty can be quantified and explained. Future work will focus on pretraining strategies for TTT.

6 Limitations

Datasets for critical tasks. The datasets used in this pilot study are not related to critical tasks. Thus, it would be useful that the scientific community shares tabular/text datasets for medical or financial applications in order to test the efficacy of the approach presented here.

Overall attention with unbalanced sequences. The effect of having sequences with different lengths (e.g. long text sequence and few tabular features) has not been assessed. Therefore, the impact of unbalanced lengths for text and tabular sequences remains to be studied.

Uncertainty quantification and coverage. The size of the prediction sets generated by TTT's streams is bounded by two. Unlike conformal prediction, it is not possible to specify the desired coverage level (i.e. the proportion of true label values that lie in the prediction sets). The actual coverage depends on the use case and can only be assessed on a validation dataset. Further, the conditional coverage could vary across different dimensions. For instance, it could be different across different population subgroups. Therefore, it is essential to monitor conditional coverage across these dimensions when the true labels are obtained.

7 Ethical Considerations

We presented a new Transformer-based architecture for classification tasks based on tabular datasets with text fields. Our method does not aim to predict or exploit any personal or sensitive information. We do not expect any significant risks related to this multimodal approach.

Given it is based on pretrained models and low-dimensional embeddings, the environmental impact of our study is limited. We reported the computational cost for each model in Appendix I. A large-scale test could benefit from employing specific cost-related algorithms for tuning hyper-parameters (Wu et al., 2021).

With regard to transparency and compliance issues, we conducted the experiments on publicly available datasets and implemented the models by exploiting well-recognized pretrained models and Python libraries, e.g. Pytorch (Paszke et al., 2019). The experimental results are obviously specific to these datasets. We ran some of our experiments on a sample of the *jigsaw*[1] public dataset which is

[1] https://www.kaggle.com/c/

used for detecting toxic comments. We reported the performance results, but did not display any harmful content from this dataset. Lastly, it is important to monitor the performance of TTT over time. In particular, if the use case is related to individuals, practitioners should ensure that the model does not achieve different levels of uncertainty or performance across population subgroups. Artefacts such as the ESG model card can help report these risks and limitations (Bonnier and Bosch, 2023).

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Javier Antoran, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. 2021. Getting a {clue}: A method for explaining uncertainty estimates. In *International Conference on Learning Representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Transformers for tabular data representation: A survey of models and applications. *Transactions of the Association for Computational Linguistics*, 11:227–249.

Thomas Bonnier. 2023. Leveraging error patterns to correct prediction intervals. In *ECAI 2023*, pages 287–294. IOS Press.

Thomas Bonnier and Benjamin Bosch. 2023. Towards safe machine learning lifecycles with esg model cards. In *Computer Safety, Reliability, and Security. SAFECOMP 2023 Workshops*, pages 369–381, Cham. Springer Nature Switzerland.

Leo Breiman. 2001. Random forests. *Machine learning*, 45:5–32.

Charles Corbière, Nicolas THOME, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. 2019. Addressing failure prediction by learning model confidence. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Julian Eisenschlos, Maharshi Gor, Thomas Müller, and William Cohen. 2021. MATE: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. PMLR.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2022. On embeddings for numerical features in tabular deep learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 24991–25004. Curran Associates, Inc.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, pages 18932–18943. Curran Associates, Inc.

Ken Gu and Akshay Budhkar. 2021. A package for learning on tabular and text data with transformers. In *Proceedings of the Third Workshop on Multimodal Artificial Intelligence*, pages 69–73, Mexico City, Mexico. Association for Computational Linguistics.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, LIU Shujie, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. 2020. Graphcodebert: Pre-training code representations with data flow. In *International Conference on Learning Representations*.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

`jigsaw-unintended-bias-in-toxicity-classification`

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456, Online. Association for Computational Linguistics.

Satyapriya Krishna, Tessa Han, Alex Gu, Javin Pombra, Shahin Jabbari, Steven Wu, and Himabindu Lakkaraju. 2022. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv preprint arXiv:2202.01602*.

Solomon Kullback. 1997. *Information theory and statistics*. Courier Corporation.

Ping Li, Trevor J Hastie, and Kenneth W Church. 2006. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296.

Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2022. Foundations and recent trends in multimodal machine learning: Principles, challenges, and open questions. *arXiv preprint arXiv:2209.03430*.

J. Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Cuong V Nguyen, Sanjiv R Das, John He, Shenghua Yue, Vinay Hanumaiah, Xavier Ragot, and Li Zhang. 2021. Multimodal machine learning for credit modeling. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1754–1759. IEEE.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alexander Gammerman. 2002. Inductive confidence machines for regression. In *Machine Learning: ECML 2002, 13th European Conference on Machine Learning, Helsinki, Finland, August 19-23, 2002, Proceedings*, volume 2430 of *Lecture Notes in Computer Science*, pages 345–356. Springer.

Letitia Parcalabescu and Anette Frank. 2023. MM-SHAP: A performance-agnostic metric for measuring multimodal contributions in vision and language models & tasks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4032–4059, Toronto, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Mauricio Sadinle, Jing Lei, and Larry Wasserman. 2019. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525):223–234.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

LS Shapley. 2016. 17. a value for n-person games. In *Contributions to the Theory of Games (AM-28), Volume II*, pages 307–318. Princeton University Press.

Xingjian Shi, Jonas Mueller, Nick Erickson, Mu Li, and Alex Smola. 2021. Benchmarking multimodal automl for tabular data with text fields. In *Thirty-fifth Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*.

Erik Štrumbelj and Igor Kononenko. 2010. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(1):1–18.

Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J. Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6558–6569, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde,

Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Springer Science & Business Media.

Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qingyun Wu, Chi Wang, and Silu Huang. 2021. Frugal optimization for cost-related hyperparameters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10347–10354.

Peng Xu, Xiatian Zhu, and David A. Clifton. 2023. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12113–12132.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust transformer modeling for table-text encoding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–537, Dublin, Ireland. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114, Copenhagen, Denmark. Association for Computational Linguistics.

A Appendix: Table of Contents

B Related Work

Transformers for tabular data. With regard to tabular data, most existing works on Transformers have focused on learning the structure and content of tables with pretraining strategies. Those models are usually employed for tasks such as question answering (Yin et al., 2020; Eisenschlos et al., 2021), table based fact-checking (Yang et al., 2022), or table content population (Iida et al., 2021). Such architectures leverage specific positional embeddings and row-wise/column-wise attention. Our work, however, concentrates on employing labeled tabular/text data for multimodal classification tasks. We note that few works address this objective (Badaro et al., 2023; Gu and Budhkar, 2021).

C Datasets and Sampling

All the datasets are publicly available with one of these licenses: "CC0: Public Domain", "Competition Data", or "CC BY-NC-SA 4.0". These datasets can be accessed and used for the purpose of academic research. The text fields are in English.

In Table 6, we give more details on the datasets:

- airbnb[2]: the task is to predict the price range of Airbnb listings. The text fields are listing descriptions.

- cloth[3]: the goal is to classify the sentiment (represented as a class) of user reviews regarding clothing items. The text fields are customer reviews.

[2] https://www.kaggle.com/datasets/tylerx/melbourne-airbnb-open-data

[3] https://www.kaggle.com/datasets/nicapotato/womens-ecommerce-clothing-reviews

Dataset	# Train	# Test	# Num	# Cat	Text length	# Class
airbnb	4,372	1,367	27	23	264	10
cloth	14,481	4,526	2	3	102	5
jigsaw	64,000	20,000	5	1	126	2
kick	69,196	21,624	3	3	32	2
petfinder	9,587	2,997	5	14	135	5
salary	12,672	3,961	1	2	34	6
wine10	42,537	13,294	2	2	56	10
wine100	70,905	22,158	2	2	55	100

Table 6: Information on datasets: number of samples in training and test datasets, number of numerical/categorical features, text sequence length (maximum length based on word count), number of classes.

- jigsaw[4]: the objective is to detect toxic comments. As advised by the creators of this use case, examples with target value greater or equal to 0.5 will be considered to be in the positive class (toxic). As class imbalance can make the accuracy metric irrelevant, we oversample instances with the *toxic* label to reach 20% share, while keeping only 100,000 instances from the initial 1.8 million examples in order to reduce the computational cost.
- kick[5]: the task is to predict whether a proposed project will meet its funding goal. The text fields are project descriptions.
- petfinder[6]: the goal is to predict the speed range at which a pet is adopted. The text fields are profile write-ups for the pets.
- salary[7]: the task is to predict the salary range based on data scientist job postings. The text fields are job descriptions.
- wine[8]: the goal is to predict the variety of wines. The text fields are wine tasting descriptions.

For some of the use cases, we employ the original training dataset as the test dataset does not include the true labels (competition data). In that case, we consider the training dataset as the modeling data which is then randomly split into training-validation-test subsets. For all datasets, the splits are constructed as follows: (1) The initial dataset is randomly split into two disjoint temporary (80% share) and test (20% share) subsets, respectively; (2) The temporary dataset is randomly split into two disjoint training (80% share) and validation (20% share) subsets, respectively.

[4] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification

[5] https://www.kaggle.com/datasets/codename007/funding-successful-projects

[6] https://www.kaggle.com/competitions/petfinder-adoption-prediction/data

[7] https://machinehack.com/hackathons/predict_the_data_scientists_salary_in_india_hackathon/overview

[8] https://www.kaggle.com/datasets/zynicide/wine-reviews

D Pre-processing

Feature engineering. When the dataset contains several text fields, these are concatenated in order to obtain a single field. Rows with missing values are dropped. Missing values for categorical variables could be handled by specific embeddings, but this remains to be studied. The list of final features for each dataset is available in the *settings.py* Python file. We mention here additional features that were created from the raw dataset (detailed in *dataset.py*):

- airbnb: for this dataset only, we discretize the target variable by employing quantile binning (ten intervals with equal share of data). We also create two new features *host_since_year* and *last_review_year* by extracting the year from *host_since* and *last_review* respectively.
- kick: we compute the duration to launch (in days) with *deadline* and *launched_at*. We also log-transform *goal*.
- wine: we extract the *year* from *title*.

Tabular feature encoding. For numerical features, TTT and TTT-SRP use the distance-to-quantile embedding scheme with an arbitrary value of 6 quantiles $(0., 0.2, 0.4, 0.6, 0.8, 1.)$. The embeddings of the baselines are constructed with linear functions (EarlyConcat, MulT, LateFuse-BERT) or features are kept as standardized values (LateFuse, TFN). A linear function applies the following transformation to a scalar feature value

$x \in \mathbb{R}$: $x.W_{num} + b$ where $W_{num} \in \mathbb{R}^d$ and the bias $b \in \mathbb{R}^d$. For categorical features, we use the one-hot-encoding technique (LateFuse, TFN) or encode them as category embeddings (EarlyConcat, MulT, TTT, TTT-SRP, LateFuseBERT). In that latter case, the corresponding embedding is computed as $e^T W_{cat}$ where $e \in \mathbb{R}^{n_c \times 1}$ is a one-hot-vector for the associated categorical feature, n_c denote the number of categories for this feature, and $W_{cat} \in \mathbb{R}^{n_c \times d}$. Lastly, in AllTextBERT, the tabular features, converted to strings, and the text fields are concatenated and input into DistilBERT-base-uncased.

Text pre-processing for small architectures (EarlyConcat, LateFuse, MulT, TFN, TTT). We keep only words and whitespaces (e.g. we remove numbers and punctuation). We use the *basic_english* tokenizer from PyTorch which performs lowercasing and basic text normalization for English words. This tokenizer returns a list of tokens after splitting on whitespace. To reduce the dictionary's size, we remove words that appear only once. For the text sequence maximum length (see Table 6), the value is set as the 0.9 quantile of the text field lengths' distribution, based on words. We also use the [UNK] token for unknown words, truncation, and padding to the specific maximum length.

Text pre-processing for large architectures (AllTextBERT, LateFuseBERT, TTT-SRP). For these models, we perform the following text pre-processing: we keep words, numbers, and whitespaces. We then use the DistilBERT-base-uncased tokenizer based on WordPiece. For the text sequence length, the value is the 0.9 quantile of the text field lengths' distribution. We use truncation and padding to the fixed maximum length.

Key padding mask. We use key padding masks in order to specify which text tokens should be ignored (i.e. "padding") for the purpose of attention.

Hyper-parameter	Space
Embedding dimension	$\{32, 48, 64\}$
Layers (Transformer)	$\{2, 3\}$
Heads (Transformer)	$\{4, 8\}$
Learning rate	LogUniform$[1e-5, 1e-3]$
Batch size	$\{32, 64, 96, 128\}$

Table 8: Hyper-parameter space used during tuning with Optuna (small architectures).

Hyper-parameter	Space
Embedding dimension	$\{120, 144, 168, 192, 216, 240\}$
Layers (Transformer)	$\{4, 5\}$
Heads (Transformer)	$\{4, 8, 12\}$
Learning rate	LogUniform$[1e-5, 1e-3]$
Batch size	$\{32, 64, 96, 128\}$

Table 9: Hyper-parameter space used during tuning with Optuna (TTT-SRP).

E Hyper-parameter Tuning, Weight Initialization, and Training Settings

Small architectures. For all the small architectures (EarlyConcat, LateFuse, MulT, TFN, TTT), the hyper-parameters are tuned on the validation dataset. The objective is to select the best trial, i.e. the trial with the highest accuracy on the validation subset. We run a Bayesian Optimization algorithm (Tree-structured Parzen Estimator) by using Optuna library (Akiba et al., 2019) with a limited time budget for each model/run: 900s for each case where the training dataset contains more than 20,000 samples; 600s otherwise. Table 8 displays the hyper-parameter space used during tuning. We also use the pruning option which enables to early-stop unpromising trials. For faster epochs, we use only 60% of the training samples if the training dataset contains more than 20,000 samples.

The selected model is then trained (on the training dataset) by using early stopping with a patience of 4 for the accuracy on the validation set. An exponential learning rate scheduler with gamma of 0.9 (multiplicative factor of learning rate decay) is also employed. We keep the best model in terms of epochs, i.e. with the highest accuracy on validation data.

Additional information on training settings is provided here: (1) We use the Xavier uniform weight initialization (Glorot and Bengio, 2010) for the parameters in the attention modules and the Kaiming uniform weight initialization (He et al., 2015) for the other weights; (2) Loss: cross-entropy; (3) Loss optimization: optimizer (Adam) with weight decay ($1e-5$); (4) Maximum number of epochs: 100; (5) Dropout rate (for attention, residual, feed-forward network, fully-connected networks, embeddings): 0.1.

TTT-SRP. For TTT-SRP, the hyper-parameters are tuned on the validation dataset. The objective is to select the best trial, i.e. the trial with the highest accuracy on the validation subset. We run a Bayesian Optimization algorithm (Tree-structured

Model	airbnb	cloth	jigsaw	kick	petfinder	salary	wine10	wine100
EarlyConcat	0.005	0.005	0.005	0.001	0.012	0.010	0.002	0.004
LateFuse	0.015	0.005	0.003	0.027	0.015	0.007	0.008	0.004
MulT	0.010	0.006	0.005	0.003	0.009	0.022	0.011	0.008
TFN	0.017	0.032	0.003	0.004	0.007	0.006	0.002	0.031
TTT (ours)	0.011	0.007	0.003	0.002	0.013	0.010	0.006	0.004
AllTextBERT	0.037	0.003	0.003	0.004	0.012	0.009	0.006	0.008
LateFuseBERT	0.029	0.006	0.004	0.009	0.012	0.004	0.005	0.006
TTT-SRP (ours)	0.009	0.010	0.002	0.002	0.011	0.010	0.007	0.006
Ablation 1	0.023	0.009	0.004	0.002	0.010	0.009	0.006	0.004
Ablation 2	0.019	0.009	0.002	0.005	0.012	0.007	0.006	0.008
Ablation 3	0.014	0.005	0.005	0.002	0.010	0.010	0.006	0.005

Table 7: Standard deviation of the accuracy on the test dataset computed over 5 random seeds.

Parzen Estimator) by using Optuna library with a limited time budget for each model/run: 900s for each case where the training dataset contains more than 20,000 samples; 600s otherwise. Table 9 displays the hyper-parameter space used during tuning. We also use the pruning option which enables to early-stop unpromising trials. For faster epochs, we use only 60% of the training samples if the training dataset contains more than 20,000 samples.

The selected model is then trained (on the training dataset) by using early stopping with a patience of 1 for the accuracy on the validation set. An exponential learning rate scheduler with gamma of 0.9 (multiplicative factor of learning rate decay) is also employed. We keep the best model in terms of epochs, i.e. with the highest accuracy on validation data.

The input embeddings are initialized with pretrained DistilBERT-base-uncased input embeddings after reducing their dimensions through sparse random projections: $E = \frac{1}{\sqrt{d}}AR$, where $E \in \mathbb{R}^{v \times d}$ denotes the reduced input embeddings from TTT-SRP ($v = 30522$ is the vocabulary size and d is the embedding dimension selected through hyper-parameter tuning), $A \in \mathbb{R}^{v \times D_{DBERT}}$ is DistilBERT input embeddings ($D_{DBERT} = 768$ is the embedding dimension), and $R \in \mathbb{R}^{D_{DBERT} \times d}$ is the projection matrix with elements r_{ij}. As per (Li et al., 2006), we have $r_{ij} = \sqrt{s}$ with probability $\frac{1}{2s}$, $r_{ij} = 0$ with probability $1 - \frac{1}{s}$, and $r_{ij} = -\sqrt{s}$ with probability $\frac{1}{2s}$. As advised by the authors, we take $s = \sqrt{D_{DBERT}}$. Further, we use the Xavier uniform weight initialization for the parameters in the attention modules and the Kaiming uniform weight initialization for the remaining weights.

Lastly, the other elements are similar to small architectures: (1) Loss: cross-entropy; (2) Loss optimization: optimizer (Adam) with weight decay ($1e-5$); (3) Maximum number of epochs: 100; (4) Dropout rate (for attention, residual, feed-forward network, fully-connected networks, embeddings): 0.1.

dataset	val cov.	test cov.	ag. cov.	dis. cov.
airbnb	0.52	0.51	0.49	0.52
cloth	0.73	0.72	0.72	0.72
jigsaw	0.91	0.91	0.91	0.93
kick	0.89	0.89	0.89	0.92
petfinder	0.50	0.51	0.49	0.52
salary	0.57	0.57	0.56	0.60
wine10	0.85	0.85	0.86	0.84
wine100	0.72	0.72	0.73	0.68

Table 10: TTT coverage on the validation (val cov.) and test (test cov.) datasets. Coverage when the streams of TTT agree versus when they disagree on test data, averaged over 5 seeds.

AllTextBERT, LateFuseBERT. In order to finetune these pretrained models, we follow the guidelines from Devlin et al. (2019); Sanh et al. (2019) with a batch size of 32 and a learning rate of $5e-5$. The model is trained (i.e. fine-tuned on the training dataset) by using early stopping with a patience of 1 for the accuracy on the validation set. An exponential learning rate scheduler with gamma of 0.9 (multiplicative factor of learning rate decay) is also employed. We keep the best model in terms of epochs, i.e. with the highest accuracy on validation data.

For the tabular Transformer component of LateFuseBERT, we use the Xavier uniform weight initialization for the parameters in the attention modules and the Kaiming uniform weight initialization for the other weights.

Lastly, we also have: (1) Loss: cross-entropy; (2) Loss optimization: optimizer (AdamW) with weight decay ($1e-5$); (3) Maximum number of

epochs: 100; (4) Dropout rate: 0.1.

F Variability in Results

Table 7 displays the variability in performance.

G Additional Baseline

We experiment with a different type of baseline: a random forest (100 estimators, maximum depth of 10) based on term frequency-inverse document frequency (TF-IDF) with 1000 text features, one-hot-encoding for the categorical features, and the scaled numerical features. This model achieves the same mean accuracy as TTT on 2 small datasets with many tabular features (airbnb and petfinder). However, its accuracy is weaker on the other datasets: airbnb 0.385 | cloth 0.555 | jigsaw 0.806 | kick 0.685 | pet 0.391 | salary 0.372 | wine10 0.67 | wine100 0.468.

H Uncertainty Quantification

LAC baseline. The conformity score of the LAC method (Least Ambiguous Set-Valued Classifiers) from (Sadinle et al., 2019) is one minus the probability of the true label. Thus, we compute it for each data point of the validation dataset (size n). To make the methods comparable, the target coverage level $(1-\alpha)$ is set to the one obtained on the validation set with the stream disagreement (SD) method. We then compute the $(1-\alpha)(n+1)/n$ quantile of the conformity scores's distribution. For each data point of the test dataset, the prediction set is then constructed with the labels where one minus the predicted probability is below the previous quantile.

MCD baseline. For each data point of the test dataset, we obtain $M = 50$ softmax distributions with the Monte Carlo dropout method. We then estimate the uncertainty at each datapoint by computing the entropy u as in (Antoran et al., 2021): $u = -\sum_{c\in\mathbb{Y}}(\frac{1}{M}\sum_{i=1}^{M} P_{i,c})\log_2(\frac{1}{M}\sum_{i=1}^{M} P_{i,c})$, where $P_{i,c} = \hat{p}(c|\theta_i, x)$ is the predicted probability from TTT for simulation i, model parameters θ_i, and class c.

Coverage. The prediction regions produced by TTT can be more informative than a single predicted label. For example, TTT's accuracy is 47% on *salary* dataset, but Table 10 shows that the prediction sets achieve 57% coverage and 35% of the sets contain two predicted labels. From a frequentist standpoint, if we consider the whole test dataset, 57% of the time, the prediction sets of size 1 or 2 would include the true label. Lastly, Table 10 also shows the coverage achieved by TTT's prediction sets when the two streams agree or disagree. These coverage values are not equal but remain quite close.

Example. Lastly, Figure 4 displays two examples where the streams disagree and the model fails to predict the correct label.

I Implementation Information

Hardware and computational cost. We run the experiments with a Tesla T4 GPU. Table 11 summarizes the average computational cost for each model. It is worth noting that the computation time for TTT-SRP includes hyper-parameter tuning (between 600 and 900s), which explains why the total computation time is as high as LateFuseBERT's one.

Model	Time (s)	Parameters
EarlyConcat	933	1.2M
LateFuse	910	1.2M
MulT	1216	1.3M
TFN	881	1.3M
TTT	1095	1.3M
AllTextBERT	1748	66.4M
LateFuseBERT	1178	81.3M
TTT-SRP	1179	8.2M

Table 11: Average computation time (in seconds) per run and average number of model parameters. This includes pre-processing, hyper-parameter tuning, training (or fine-tuning for pretrained models), and evaluation.

Python libraries. The implementation is based on Python 3.10 and the following packages: torch 2.1.0+cu118 (Paszke et al., 2019), optuna 3.4.0 (Akiba et al., 2019), transformers 4.35.2 (Wolf et al., 2020), pandas 1.5.3 (Wes McKinney, 2010), numpy 1.23.5 (Harris et al., 2020), sklearn 1.2.2 (Pedregosa et al., 2011), scipy 1.11.4 (Virtanen et al., 2020). These libraries are publicly available with "BSD", "MIT", or "Apache Software" licenses.

For AllTextBERT and LateFuseBERT, we use the *'distilbert-base-uncased'* tokenizer and model from the transformers library. All the baselines are implemented with Pytorch and adapted to tabular-text inputs. The code has been written based on the content of the references cited in Section 4.1.

Division Name	Department Name	Class Name	Age	Positive Feedback Count	Review	Y	Final pred	text_pred	tabular_pred
General Petite	Tops	Knits	39	1	Nice design Love the rushed sleeves, and the flower pattern on the side. t shirt is, however, a little thin	4	3	3	4
General	Jackets	Outerwear	37	4	Beautiful colors but poor quality and bulky I was excited to receive this coat for my annual travel to switzerland	1	0	0	1

	CLS-tab	Division Name: General Petite	Department Name: Tops	Class Name: Knits	Age: 39	Positive Feedback Count: 1	CLS-text	nice	design	love	the	rushed	sleeves	and	the	flower	pattern	on	the	side	t	shirt	is	however	a	little	thin
CLS-tab	0.012	0.006	0.011	0.006	0.036	0.003	0.004	0.277	0.106	0.043	0.007	0.007	0.003	0.004	0.004	0.050	0.006	0.017	0.002	0.014	0.011	0.002	0	0.249	0.001	0.041	0.074

	CLS-tab	Division Name: General	Department Name: Jackets	Class Name: Outerwear	Age: 37	Positive Feedback Count: 4	CLS-text	beautiful	colors	but	poor	quality	and	bulky	i	was	excited	to	receive	this	coat	for	my	annual	travel	to	switzerland
CLS-tab	0.008	0.001	0.006	0.009	0.012	0.006	0.003	0.048	0.009	0.289	0.420	0.024	0.004	0.044	0.002	0.013	0.045	0.013	0.003	0.005	0.006	0.004	0.005	0.001	0.007	0.011	0.001

Figure 4: Illustration of predictive uncertainty with TTT for two instances of *cloth* dataset. *Top*: Table with inputs, true label, predicted label, prediction from the text stream, and prediction from the tabular stream. In each example, the model is uncertain and hesitates between two labels. The model fails to predict the correct label as it follows the prediction of the wrong stream. *Bottom*: The attention weights of $[CLS]_{tab}$ averaged across heads are displayed for the first layer of the model and for each example.

J Models' Architectures

TTT and TTT-SRP. We detail here the feed-forward module and final fully-connected layers from Figure 2. The feed-forward module can be described as follows: $\text{FF}(x) = \text{Linear}(\text{Dropout}(\text{ReLU}(\text{Linear}(x))))$ where the output has an embedding dimension of d.

The final fully-connected layers can be described as follows: $\text{FC}(x) = \text{Linear}(\text{Dropout}(\text{ReLU}(\text{Linear}(x))))$ where the output has a dimension of C (number of classes).

TTT-SRP's architecture is quite similar. As previously detailed, the only differences are related to the weight initialization scheme of the input embeddings and the size of the network.

Baseline 1: EarlyConcat. The model is a one-stream Transformer with self-attention applied on the concatenated tabular-text embedding sequence (including [CLS] token). Positional embeddings are used too. The final prediction is performed after projecting the embeddings of the [CLS] token through fully-connected layers.

Baseline 2: LateFuse. Only the text stream goes though a Transformer model with self-attention. The [CLS] output of this Transformer is then concatenated with the standardized numerical features and one-hot-encoded categorical variables. The final prediction is performed after projecting this vector through fully-connected layers.

Baseline 3: MulT. This model has two streams: tabular and text. Each stream contains a Transformer with cross-modal attention followed by a Transformer with self-attention. Positional embeddings are used in this architecture too. We then concatenate the [CLS] text and tabular embeddings from each stream in order to project the resulting vector through fully-connected layers.

Baseline 4: TFN. This is the only model which is not based on a Transformer architecture. This model starts with two streams: an LSTM for the text sequence and a neural network for the tabular features. We compute the Cartesian product from the outputs of these two networks (after adding 1 in each embedding vector) in order to take into account 1-D and 2-D effects. The resulting matrix is flattened and projected through fully connected layers.

Baseline 5: AllTextBERT. The tabular features, converted to strings, and the text fields are concatenated and input into DistilBERT. The final prediction is performed after projecting the embeddings of the [CLS] token through fully-connected layers.

Baseline 6: LateFuseBERT. This model has two streams: tabular and text. Each stream contains a Transformer architecture: DistilBERT is used for the text stream and a Transformer with 3 layers (self-attention with 8 heads) is employed for the tabular stream. We then concatenate the [CLS] text and tabular embeddings from each stream in order to project the resulting vector through fully-connected layers.

K Explaining the Uncertainty

Contribution of special token [CLS]. When we compute the token contributions, the classification token [CLS] is not considered. This token is only used for the classification task.

Lasso parameters in Kernel SHAP. When computing the Kernel SHAP contributions, we fit a weighted Lasso regression with the following settings: L1 regularization penalty: 1e-3, maximum number of iterations: 1000, tolerance for the optimization: 0.0001.

Example 1: OOD instance from *wine*. The original *wine* dataset contains more than 100 classes. Thus, we select an instance from a minority class which is not included in *wine100*. The uncertain OOD instance has the following characteristics:

- Tabular features: *country*="Italy", *year*="2005", *price*=75.0, *points*=91.
- Text: "This floral wine is bursting with notes of jasmine and honeysuckle backed by sweet fragrances of candied fruit and caramel. The wine is sweet and thick in the mouth but the well-dosed acidity keeps it from being cloying or too fat. It would pair well with slightly sweet foods such as honey glazed ham or lobster."
- Label and predictions: $y = 611$, $\hat{y}_{text} = 21$, $\hat{y}_{tab} = 43$, $\hat{y} = 97$.
- $JSD(p, q) = 0.82$, $\hat{f}((p, q)) = 0.84$.

The feature contributions to uncertainty are displayed in Figure 5. There are several positive (red) contributions, proving that the input may be unusual. The main source of uncertainty is related to the country along with specific words. This method might be used for novelty detection as well. In that case, using the uncertainty classifier produces more reliable outputs as confirmed by the Pearson correlations between the feature contributions generated by our method and by Kernel SHAP: 0.68 (JSD) and 0.89 (uncertainty classifier).

Example 2: uncertain instance from *cloth*. In Figure 6 (first 2 charts), the distributions of the JSD metric and uncertainty classifier's predicted probability evidence that the uncertainty classifier can better distinguish between certain and uncertain instances.

The uncertain instance has the following characteristics:

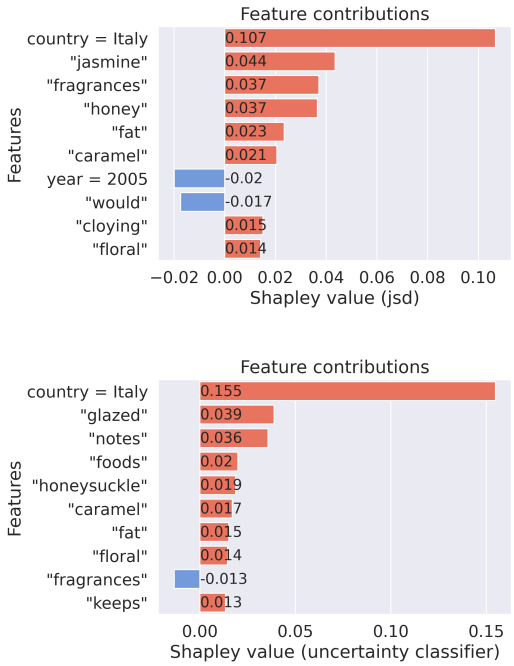

Figure 5: Example 1. The top 10 feature contributions to uncertainty for an OOD instance from *wine* dataset. Top: with JSD value function. Bottom: with uncertainty classifier.

- Tabular features: *Division Name*="General", *Department Name*="Bottoms", *Class Name*="Shorts", *Age*=43, *Positive Feedback Count*=0.
- Text : "Romper rules Overall great romper! i have a long torso and thicker legs so i sized up to a small and i am very happy i did. the shorts are a bit on the shorter side and go upon the sides so another reason to consider sizing up. the quality is great and super soft chambray with a trendy tie up."
- Label and predictions: $y = 3$, $\hat{y}_{text} = 4$, $\hat{y}_{tab} = 3$, $\hat{y} = 4$.
- $JSD(p, q) = 0.81$, $\hat{f}((p, q)) = 1$.

The feature contributions to uncertainty are displayed in the last two rows of Figure 6. The model's decision is incorrect. This is the same prediction as the output of the text stream, whereas the tabular stream predicts the correct label. The uncertainty mainly originates from words (e.g. *overall*, *thicker*). In that case, the explanations generated with each value function are quite consistent. Further, the Pearson correlations between the feature

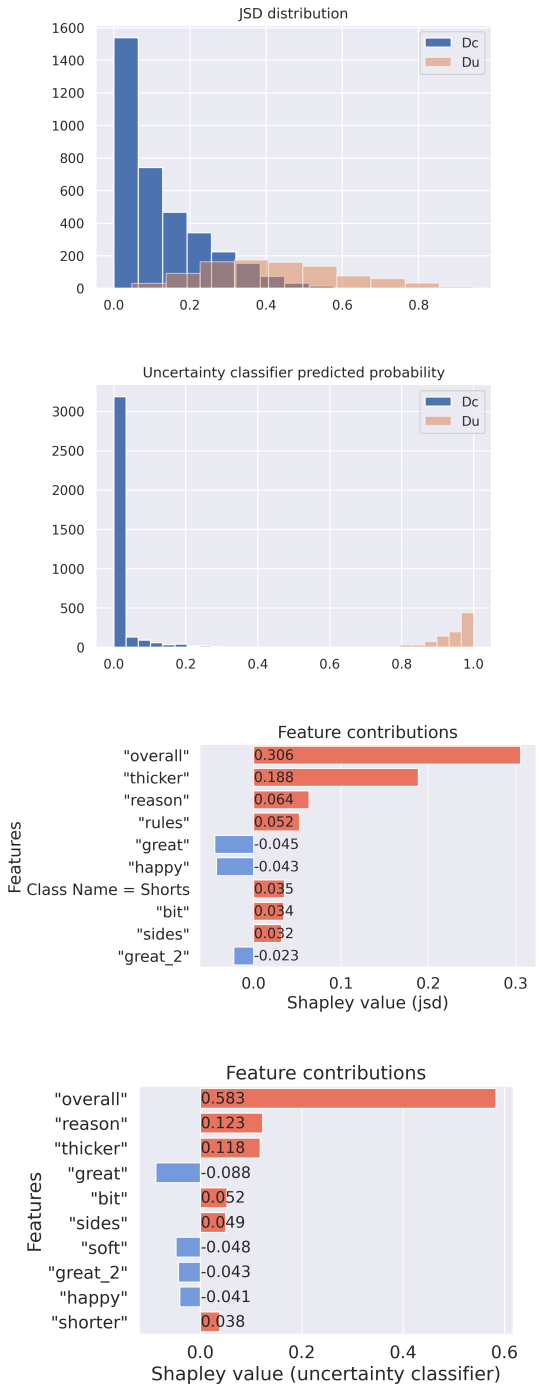

Figure 6: Example 2 on *cloth* dataset. First two rows: Distributions of value functions' outputs by subset D_c and D_u, for JSD and the uncertainty classifier, respectively. Last two rows: the top 10 feature contributions to uncertainty for an uncertain instance. Top: with JSD value function. Bottom: with uncertainty classifier.

contributions generated by our method and by Kernel SHAP is high for each value function: 0.90 (JSD) and 0.93 (uncertainty classifier).

Example 3: uncertain instance from *kick*. In Figure 7 (first 2 charts), the distributions of the JSD metric and the uncertainty classifier's predicted probability evidence that the uncertainty classifier can better discriminate between certain and uncertain instances.

The uncertain instance has the following characteristics:

- Tabular features: *country*="US", *currency*="USD", *disable communication*="False", *log goal*=4, *backers count*=447.
- Text : "The Life & Times of a Remarkable Misfit 18 months ago, I canceled a book deal with a major publisher because I realized I had sold out to be chosen. Today I choose me."
- Label and predictions: $y = 1$, $\hat{y}_{text} = 0$, $\hat{y}_{tab} = 1$, $\hat{y} = 0$.
- $JSD(p, q) = 0.27$, $\hat{f}((p, q)) = 1$.

The task is to predict whether a proposed project will meet its funding goal. The feature contributions to uncertainty are displayed in the last two rows of Figure 7. If we consider the uncertainty classifier value function, the inconsistency between the high value of *backers count* and the word *canceled* seems to make the model's decision uncertain. The Pearson correlations between the feature contributions generated by our method and by Kernel SHAP is high for each value function: 0.96 (JSD) and 0.99 (uncertainty classifier).

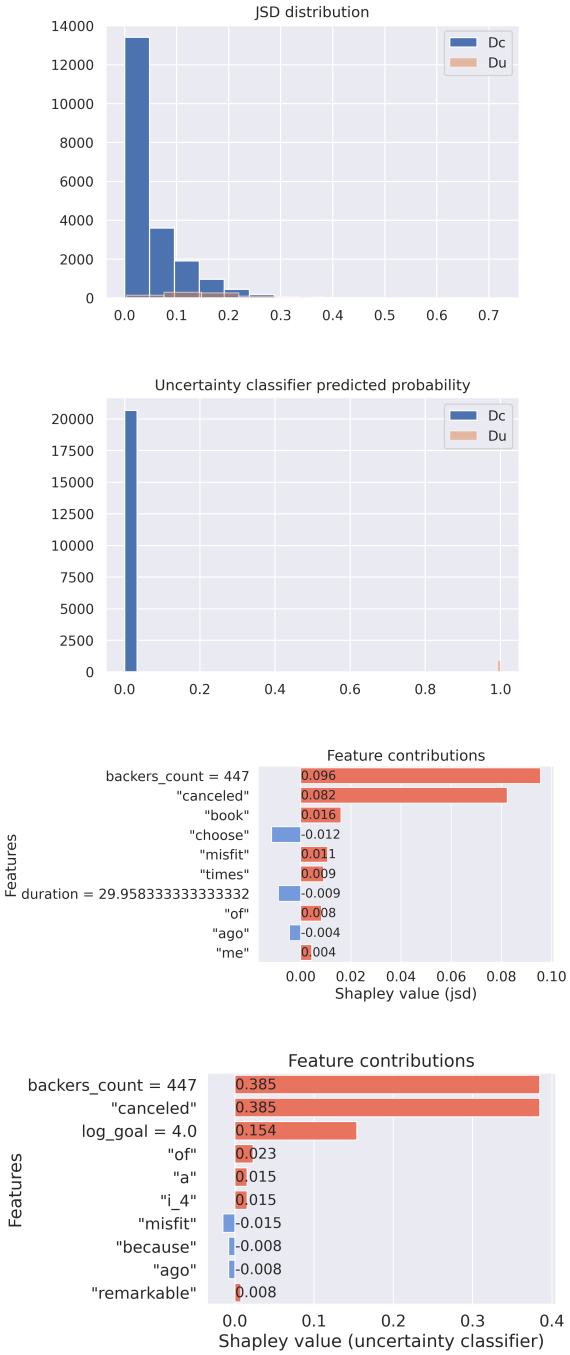

Figure 7: Example 3 on *kick* dataset. First two rows: Distributions of value functions' outputs by subset D_c and D_u, for JSD and the uncertainty classifier, respectively. Last two rows: the top 10 feature contributions to uncertainty for an uncertain instance. Top: with JSD value function. Bottom: with uncertainty classifier.