Enhancing Job Evaluation with Data Augmentation and Text Classification

Samaneh Jalilian[1], Niels van Weeren[1,2], Mohammad Shokri[1], Thijmen Bijl[1], Suzan Verberne[2]
[1]Randstad [2]Leiden University
{samaneh.jalilian, niels.van.weeren, mohammad.shokri, thijmen.bijl}@randstad.com
s.verberne@liacs.leidenuniv.nl

Abstract

Accurate job grading and evaluation are essential for ensuring fair compensation in Human Resources (HR) planning. In this research, we propose to improve job evaluation by semi-automating a manual, time-consuming, and inconsistent process with text-based classification models. We address three prediction tasks: job title classification, grading, and compensation prediction. For job title classification, we fine-tune a RoBERTa model for classification and use Gemini to generate synthetic job descriptions for rare job titles. For grade and compensation prediction, we compare TF-IDF and transformer-based embeddings. We optimize all models using grid search with hyperparameter tuning and cross-validation. The results show that job title classification by RoBERTa with Gemini-generated descriptions works well with an accuracy of about 97%. In our regression experiments, our models get promising results: for grade prediction, a tuned TF-IDF + XGBoost model achieves a mean absolute error (MAE) of 0.185, and for annual salary prediction, MiniLM embeddings with XGBoost get an MAE of €1,587. These findings demonstrate that a semi-automated pipeline can enhance traditional manual processes by boosting consistency, speeding up HR workflows, and reducing biased assessments.

1 Introduction

In today's competitive labor market, a challenge is evaluating job roles, grading them, and setting appropriate compensation. Recruiters rely on job titles, role descriptions, and responsibility levels to determine job grades and salary structures. These steps are crucial for attracting candidates and keeping current employees. Traditional methods, like the Hay Method (Hay, 1984), Mercer's IPE[1], and WTW[2], evaluate jobs by assigning weights and labels based on responsibilities, knowledge, and skills. These approaches are slow, costly, and vulnerable to bias, which can cause inconsistencies in the evaluation process. With semi-automated methods, it is possible to reduce bias in salary setting and performance reviews, helping make compensation more fair and transparent (Haseeb et al., 2024). These methods also support collecting and analyzing employee data for decisions and workforce planning (Arora et al., 2021). This can offer businesses a significant advantage in strategic compensation planning. However, challenges such as the quality and diversity of training data, incomplete datasets, and the lack of continuous model evaluation and updates can result in biased models (Mujtaba and Mahapatra, 2024). We draw on the Mincer earnings equation (Mincer, 1958) to capture how schooling and work experience affect setting compensation. We evaluate multiple text representation methods, including transformer embeddings (DistilRoBERTa, MPNet, MiniLM), TF-IDF vectorization, and their combinations with several machine learning models (Random Forest, XGBoost, and Deep Neural Networks), to identify the best feature extraction strategy for compensation prediction.

In summary, we make four contributions: (1) We propose a semi-automated job evaluation pipeline that combines transformer-based classification with LLM-generated synthetic job descriptions to improve models and address class imbalance in rare job title prediction. (2) We benchmark traditional text representations (TF–IDF) against transformer-based embeddings (RoBERTa, MPNet, MiniLM) combined with deep neural networks and tree-based models for grade and compensation prediction. (3) We show that LLM-based augmen-

[1] https://www.mercer.com/solutions/talent-and-rewards/job-architecture/job-evaluation-ipe/

[2] https://www.wtwco.com/-/media/wtw/solutions/services/job-levelling-solutions.pdf

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 872–883

tation substantially improves job title classification for rare classes. (4) Our experimental results show that surprisingly, TF-IDF features outperform transformer-based embeddings for grade prediction, while for salary prediction, transformer-based embeddings achieve higher accuracy than TF-IDF.

For reproducibility, we provide a code repository at `https://github.com/sjalilian/Jobeval-system.git`

2 Related work

Economic earnings models offer various ways to understand how wages form. The Mincer earnings function shows that each extra year of education raises earnings by about 8% (Mincer, 1958). Hedonic wage models explain compensation variation using job attributes (Rosen, 1974) and tournament theory then explains reward structures like bonuses or promotions in competitive labor systems to encourage employees to work harder (Lazear and Rosen, 1981). These economic models show that compensation depends on multiple factors.

Natural Language Processing (NLP) and Machine Learning (ML) models can enhance job description analysis, ensuring better alignment between job postings, candidate qualifications, and salary, optimizing recruitment processes (Arora et al., 2021). Recent research shows that embedding-based models can improve the quality of both classification and regression models in HR (Silva et al., 2025). Pias et al. (2024) proposes a recommendation system that matches candidate resumes (job seeker profiles) with job descriptions using TF-IDF, Cosine similarity, Jaccard similarity, and BERT embeddings. Their evaluation on a dataset of 10,000 job postings shows that BERT outperforms traditional similarity measures, achieving an accuracy of 98%. Maitra et al. (2024) develop an improved BERT model to classify job titles from job descriptions. Their model achieves 84% accuracy and an F1-score of 83% on a dataset of more than 12,000 cleaned job ads, outperforming DistilBERT and all machine learning baselines. Recent studies show that using LLM-generated data can improve HR models by enriching training sets with realistic and diverse job-related examples. Some approaches have enhanced skill matching, knowledge extraction, and recommendation tasks, helping HR systems work better (Magron et al., 2024; Wasi, 2024).

For compensation prediction, some research has shown the power of machine learning in job evaluation (Wang, 2022; Matbouli and Alghamdi, 2022). Ensemble learning methods like tree-based regressions have shown strong performance in predicting suitable salaries in complex job market datasets (Dutta et al., 2018; Anderson et al., 2024). In another research, Deng et al. (2018) compare a three-layer deep neural network (with RMSProp optimization) against SVM, Random Forest, and XGBoost baselines. They find that the DNN outperforms all tree-based models in both micro and macro precision in salary prediction.

In this paper, we show how model effectiveness depends on the prediction task, revealing new insights into compensation modeling.

3 Data

The dataset provided by a global HR organization contains a wide range of job postings in Dutch and English. These job postings were originally sourced from public job advertisements that companies posted to recruit for their open positions. The data does not include personal data, CVs, or individual-level information. The objective of this research is to estimate a job title, salary, and job grade for a job posting before publication, rather than to determine the salary of a specific individual based on their experience or background. The dataset includes 235,277 job records and 60 features. Each row in the dataset includes a job description and features such as job titles, required skills, job domains, education, specializations, reference scores, assigned job grades, and salaries. The format of the data was inconsistent (e.g., commas in salary values). After cleaning, preprocessing, and standardizing the dataset, it was prepared for model training. The dataset is not publicly available due to privacy and proprietary restrictions imposed by the data provider. Appendix A shows the distribution of the grade and salary values in the data.

4 Methods

Our methods include: (1) job title classification, (2) grade and salary prediction. Figure 1 provides a detailed flowchart of our methods.

4.1 Task 1: Job Title Classification

The goal of the first task is to automatically assign a standardized job title to each job description. This replaces a manual and time-consuming process cur-

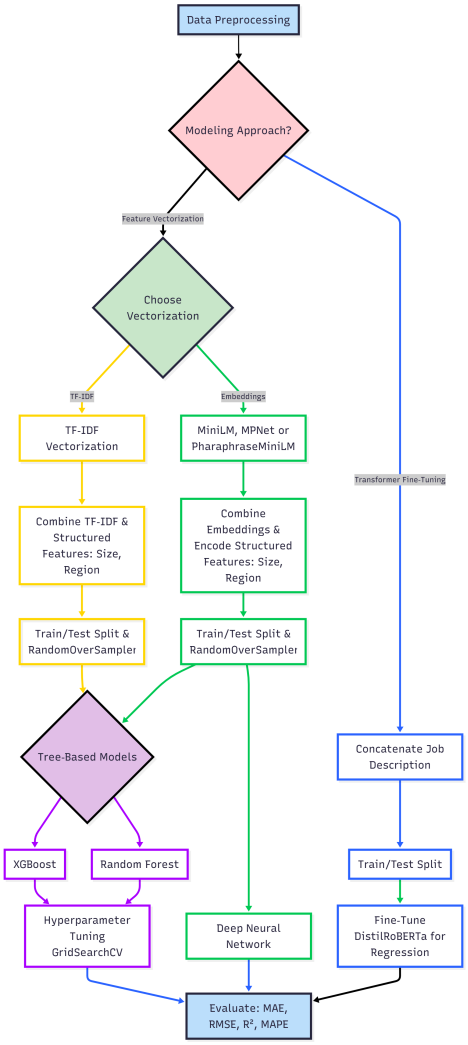

Figure 1: Methodology for Grade & Salary Prediction. The process compares two primary strategies: (1) an end-to-end approach involving the direct fine-tuning of a DistilRoBERTa model for regression, and (2) a feature vectorization approach that uses either sparse TF–IDF vectors or dense sentence embeddings from transformer models.

rently performed by HR experts. Since job titles are derived primarily from textual descriptions, we model this task as a multi-class text classification problem. To classify job descriptions into standardized job titles, we use transformer-based embeddings as text representations because they capture the semantic meaning of words in context. This part is crucial for distinguishing between hundreds of job titles that often have overlapping keywords. Each job description is tokenized using Hugging Face's `RobertaTokenizerFast`, where text is truncated or padded to a maximum sequence length of 512 tokens, and then converted into input IDs and attention masks.

Data augmentation To address the class imbalance issue in job title distribution, we apply data augmentation using Gemini (Gemini Team, 2025), a large language model, to generate synthetic job postings for rare job titles. The augmented and original job descriptions are tokenized and converted into input IDs and attention masks, which are directly fed into a RoBERTa model for multi-class job title prediction. The model generates contextual embeddings internally, and the final representation is passed to a classification head for prediction. We fine-tune the entire model end-to-end using AdamW, with tuned hyperparameters such as batch size and number of epochs. Details of the model training are listed in Appendix C.

4.2 Task 2: Grade & Salary prediction

4.2.1 Feature Extraction

TF-IDF We experiment with TF–IDF weighted word features[3] because of their interpretability and strong performance on sparse text classification problems (Romadon et al., 2020). Before vectorization, all job descriptions are preprocessed through lowercasing and punctuation removal. To capture short linguistic patterns, we include both unigrams and bigrams, and restrict the vocabulary to the 2000 most informative terms based on document frequency. Sublinear term frequency scaling is applied to reduce the influence of highly repetitive words.

Embedding We evaluate dense sentence embeddings generated by pretrained transformer encoders from the Sentence-BERT (SBERT) framework (Reimers and Gurevych, 2019).[4]

4.2.2 Model Training

Transformer Regression In this experiment, we fine-tune a DistilRoBERTa model (6 layers, 12 attention heads, hidden size 768) with a dropout of 0.2 and a single linear head on the pooled [CLS] vector to predict grade and salary from job descriptions.

Deep Neural Network on Embeddings For this approach, we develop a deep neural network (DNN) to predict grade and salary. The input to the model is dense sentence embeddings (e.g., MPNet,

[3]We use the `TfidfVectorizer` from scikit-learn (Pedregosa et al., 2011)

[4]We use the SentenceTransformers library`https://www.sbert.net/docs/sentence_transformer/pretrained_models.html` to load each encoder and manage pooling.

MiniLM, or paraphrase-MiniLM). The model is a feed-forward neural network consisting of an input layer, four hidden layers, and two neurons for the regression target. We use the Adam optimizer with default parameters, which combines the benefits of adaptive learning rates and momentum to achieve faster convergence (Kingma and Ba, 2015).

RandomForest & XGBoost on Embeddings This methodology uses dense vectors from transformer embedding models and then feeds to train the Random Forest and XGBoost models. This setup allows a comparison between using dense embeddings and sparse TF-IDF features.

RandomForest & XGBoost on TF-IDF Features This approach uses the TF-IDF vectors as the primary input features. These vectors concatenate with the structured features. These models are chosen for their strong performance on tabular and sparse data and their relative interpretability, which is beneficial for HR applications. The models are trained to predict grade and salary in separate experiments.

Although TF-IDF lacks semantic meaning and word order, it remains a valuable and interpretable method, especially when fast training and explainability are important. To optimize the performance of tree-based models (XGBoost and Random Forest), we tune the hyperparameters with grid search. Details of the model training are listed in Appendix C and the grid search details in Appendix D.

5 Experiments and results

5.1 Experiment 1: Job title classification

In this experiment, we try to reduce manual effort by automatically assigning all unstructured features of job descriptions to a standardized job title and using transformer-based classifiers that are augmented with LLM-generated synthetic examples.

The dataset contains 321 unique job title labels, indicating a highly multi-class problem. We encode these labels using a `LabelEncoder` and perform a stratified 60/40 train-test split to preserve class proportions. Then, each text description is tokenized. In the next step, tokenized inputs are converted into Hugging Face `Dataset` objects for efficient batching, shuffling (train), and evaluation.

We fine-tune a RoBERTa-based sequence classification model. After training, we evaluate on the test set, reporting overall accuracy as well as macro-averaged precision, recall, and F1 scores.

Prompt #1: "`Write an approximately 80-100 word job description for the role titled "{lbl}". Include key responsibilities, required qualifications, and typical experience.`"
Prompt #2: "`Look at the examples in few_shot_examples. Now it's your turn. Think step-by-step and then generate the structured string in the same format for {lbl}.`"

Table 1: Prompts used for data augmentation with Gemini. The few-shot examples are listed in Appendix B

Model	Accuracy	Precision	Recall	F1-score
RoBERTa on orig data	0.969	0.814	0.835	0.816
+ Augm Data (Prompt #1)	0.967	0.842	0.858	0.847
+ Augm Data (Prompt #2)	**0.977**	**0.895**	**0.910**	**0.894**

Table 2: Performance comparison of RoBERTa for job title classification with and without data augmentation

Data augmentation To boost the model's ability to recognize underrepresented titles, we use Gemini's API (in controlled mode) via the Google Vertex AI platform to generate synthetic job descriptions for the less frequent titles. This data augmentation helps the classifier better learn the characteristics of rare job titles, leading to improved classification performance. We experiment with two types of prompts for data augmentation that are shown in Table 1. The first is a direct and straightforward instruction prompt, while the second includes chain-of-thought (CoT) reasoning and few-shot examples.We set the Temperature for Gemini to 0.7. The dataset originally contained 235,277 samples. After cleaning and deduplication, this was reduced to 69,170 unique job descriptions. To address label sparsity, we first identified job titles with fewer than 50 samples and applied data augmentation to improve these smallest categories to at least 50 samples each.This augmentation was applied only to the training split, increasing it from 41,502 to 43,318 samples. The test set was not augmented, so the evaluation remains unbiased.

Table 2 shows that data augmentation gives an improvement over the baseline and that the CoT prompt (#2) is more effective than the direct prompt (#1).

5.2 Experiment 2: Grade & salary prediction

In this experiment, we analyze which combinations of text representation methods (TF–IDF, transformer embeddings) and machine learning models (deep learning, tree-based) give the best results for grade and salary prediction. In the first approach, we fine-tune a DistilRoBERTa model for regression to predict grade and yearly salary directly from job

descriptions. In the second modeling setup, we experiment with the following models: distilroberta-base[5], paraphrase-MiniLM-L6-v2[6], all-MiniLM-L6-v2[7] and all-mpnet-base-v2[8]. Each job description is embedded in the vector space and passed through a feed-forward neural network to predict the grade and salary. This setup allows us to evaluate both the representational power of each sentence encoder and the effectiveness of a DNN for numerical regression.

In the third modeling setup, we evaluate tree-based regression models using both semantic embeddings and sparse TF–IDF features. XGBoost and Random Forest are selected because of their strong performance on tabular and mixed-feature regression tasks. This allows us to compare deep learning models against classical machine learning baselines. Tables 3 and 4 present the final regression results for grade and salary prediction across all models. The results show that the best-performing feature representation and model combination depend on the prediction task. For grade, the TF–IDF + XGBoost model achieves the highest accuracy with a MAE of 0.185 and R^2 of 0.987. This shows that grade prediction depends on clear seniority keywords in the text. TF-IDF captures clear signals such as 'senior' or '5+ years experience'. For example, a job description with 'Senior Data Engineer' and '8+ years of experience leading a team' should have a higher grade than 'Data Engineer with 1–2 years of experience.' TF-IDF keeps these signals as separate features, which makes it easier for models like XGBoost to use them. In contrast, embedding-based models focus more on overall meaning and may smooth these signals, making it harder to distinguish between similar roles (e.g., 'Engineer' vs 'Senior Engineer'). For salary prediction, MiniLM sentence embeddings combined with XGBoost produce the best performance with a MAE of €1,587. The noticeable difference in Table 4 between DNN and tree-based models comes from how the models handle the imbalanced salary distribution. Deep architectures like DistilRoBERTa and DNN models struggle with imbalanced data. In contrast, tree-based models handle imbalance better because their splits

[5] https://huggingface.co/distilroberta-base

[6] https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L6-v2

[7] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[8] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Model	MAE	RMSE	R^2	MAPE
DistilRoBERTa Regression	0.521	0.656	0.948	0.052
MiniLM + DNN	0.511	0.667	0.934	0.043
MPNet + DNN	0.493	0.634	0.940	0.042
Paraphrase-MiniLM + DNN	0.514	0.656	0.936	0.044
MiniLM + XGBoost (embs)	0.483	0.655	0.936	0.041
MiniLM + Random Forest (embs)	0.397	0.548	0.964	0.037
TF–IDF + XGBoost (tuned)	**0.187**	**0.323**	**0.987**	**0.016**
TF–IDF + Random Forest (tuned)	0.297	0.442	0.975	0.027

Table 3: Final comparison of models for grade prediction

Model	MAE (€)	RMSE (€)	R^2	MAPE
DistilRoBERTa Regression	11,539	15,522	0.579	0.187
MiniLM + DNN	13,605	19,160	0.619	0.203
MPNet + DNN	13,559	18,963	**0.626**	0.206
Paraphrase-MiniLM + DNN	13,748	19,361	0.611	0.207
MiniLM + XGBoost (embs)	**1,587**	**6,879**	0.041	**0.024**
MiniLM + Random Forest (embs)	1,747	6,921	0.057	0.026
TF–IDF + XGBoost (tuned)	1,611	6,965	0.045	0.025
TF–IDF + Random Forest (tuned)	1,688	6,905	0.065	0.027

Table 4: Final comparison of models for salary prediction

can separate the rare or extreme cases more effectively (Grinsztajn et al., 2022; Cieslak and Chawla, 2008). Unlike the error metrics (MAE, RMSE, and MAPE), which decrease with better performance, R^2 behaves differently. Its values remain low for the tree-based models, not because their predictions are inaccurate, but because salary has extremely high variance. When the target is highly dispersed, R^2 often remains low even if the model makes accurate predictions (Kvålseth, 1985). Finally, unlike grade, salary is influenced by semantic context rather than keyword frequency alone. Factors such as job complexity, technical specialization, and industry domain are captured better by dense transformer–based embeddings than by sparse vectors.

6 Job evaluator application

Based on our experimental results, we implemented a demo application for job evaluation that was put into practice at the recruitment agency. The application provides an interactive web interface for HR professionals. It allows users to paste job descriptions and automatically predicts the job title, grade, and annual salary. For each task, the system uses the best model identified after comparison results: a RoBERTa classifier (trained with augmented data) for job title prediction, a TF–IDF + XGBoost model for grade prediction, and a SentenceTransformer + XGBoost model for salary estimation. The app also includes a feedback section where HR experts can approve or reject predictions. All inputs, predictions, and feedback are automati-

cally stored in a CSV file, which is used to identify model weaknesses and improve future versions. We collected feedback from about 30 HR specialists across Asia, Europe, and the United States who tested the application. Participants had between 3 and 20 years of experience. Most said the predicted grades were close to what they expected. They reported that the system saved time in the evaluation process, and they did not need to read the full job descriptions in detail. The model worked effectively for both Dutch and English job descriptions, which is particularly valuable for international HR specialists. Approximately 25% of salary predictions were considered lower than expected. This is because the model was trained on historical (2024) data, while HR experts compare the results with current market salaries. Also, 90% of grade predictions were accepted by HR experts. Based on their feedback, we observed that the model for salary and grade prediction performs well when job descriptions include clear signals such as 'senior', 'junior', or 'years of experience'. However, performance decreases for higher-level or more abstract roles, where descriptions are less standardized. Users mentioned that the model shows limitations in distinguishing between closely related job titles, for example, 'security consultant' vs. 'security architect', and 'Project Manager' vs. 'Product Manager'.

Since the current model predicts only one title of the list, they suggested showing the top three possible titles so they can choose the best match. Screenshots of the demo application and the feedback collection interface are shown in Appendix E. A video of the functional application is accessible at https://www.youtube.com/watch?v=-7KPs0LwUnU.

7 Discussion

Data augmentation with LLMs helps to address class imbalance and improve the classification of rare job titles, but it can introduce potential risks of bias. In our experiments, synthetic job descriptions were used only during training and were never included in the test set. Also, the data consist of job postings rather than personal or individual information. As a result, the potential impact of bias is limited, since the model does not learn or reproduce sensitive personal profiles, which could create harmful biases related to gender, age, or background. To assess the potential bias of synthetic data, we conducted two analyses. Firstly, we computed the token distributions of real and synthetic texts by representing them with TF-IDF features and measuring their similarity using Jensen–Shannon divergence (Menéndez et al., 1997), a standard method for detecting distributional differences in text data. Secondly, we compared the average token length of real and synthetic texts. The Jensen–Shannon divergence is very small (≈ 0.027), and the average token lengths are almost identical. Together, these findings suggest that the synthetic data closely matches the statistical properties of real job descriptions and does not introduce noticeable distributional bias.

In the salary prediction task, the deep learning approaches did not perform well. Both the DistilRoBERTa regression model and the DNN models built on transformer embeddings produced higher errors compared to the tree-based methods. This result suggests that transformer regression is not suitable for structured value prediction, such as job grades or salaries, where training data are noisy and imbalanced. In contrast, XGBoost and Random Forest with tf-idf features were more stable and achieved far higher accuracy, confirming that tree-based ensemble methods remain strong baselines for regression tasks in HR analytics.

8 Conclusion

We introduced a machine learning pipeline for semi-automating job evaluation in the context of HR analytics. Our pipeline combines transformer-based language models with data augmentation for job title classification and hybrid regression architectures for compensation prediction. We used synthetic job descriptions generated by a large language model (Gemini) to address class imbalance in job title classification. This augmentation strategy enabled RoBERTa to achieve over 97% accuracy, demonstrating that LLMs can effectively expand training data in HR domains. In grade prediction, the TF–IDF features with an XGBoost model achieves the highest accuracy. This indicates that lexical features play an important role in grade prediction. For the salary prediction part, models built on transformer embeddings outperformed those using TF–IDF features, showing that dense semantic representations capture more predictive information. The best model combined MPNet embeddings with XGBoost. Overall, our results show that AI methods can help automate job grading and salary prediction with high quality, reduce bias, and speed

up HR workflows. Our findings demonstrate that the choice of model architecture must be aligned with the goal of the prediction task, and simpler ensemble models can outperform deep learning for real-world challenges.

A valuable direction for future work is reinforcement learning with human-in-the-loop (RLHF). In this approach, HR experts review model outputs and provide feedback, which is used as a reward signal to adjust and improve the model's predictions. This allows the system to learn not only from data, but also from domain expertise. RLHF helps in reducing bias and increases the reliability of automated job evaluation systems (Christiano et al., 2023). We also plan to explore using small decoder-only language models with a few-shot prompting approach. This would allow us to test a simpler setup that avoids heavy preprocessing and model training, and compare it directly with our current pipeline. This will help us better understand the trade-offs between prompt-based methods and structured models, and further validate the role of classical approaches in real-world HR settings.

Limitations

Training AI models requires high-quality and unbiased data. If models are trained on biased data, the algorithms may worsen existing pay disparities instead of enhancing them (Muralidharan et al., 2024; Mujtaba and Mahapatra, 2024). To address this problem, organizations must monitor and update their models continuously. Compensation estimator frameworks, if implemented carefully, have the potential to add transparency, fairness, and speed in salary decisions.

Ethics statement

Applying AI to human resources (HR) systems comes with a few ethical concerns. One of the most important legislation in this context is the EU Artificial Intelligence Act (Regulation (EU) 2024/1689), which assigns rules for the development, deployment, and use of AI within the European Union (Parliament and of the European Union, 2024). The Act's risk-based approach divides AI applications into four groups, and HR systems are marked as 'high-risk'.

In this research, in the job classification part, we use LLMs for data augmentation, while synthetic data includes some risks. Hidden biases in AI systems can come from the training data and the algorithms. Therefore, continuous monitoring must be considered throughout the entire process of adding AI in HR applications (Muralidharan et al., 2024). Organizations should keep clear documentation of their AI models and outputs so HR professionals can review and understand them. It makes sure employees understand how the automated decisions are made and gives them a clear way to contest or accept the results. Together, these practices help in building reliable frameworks. Finally, incorporating human-in-the-loop feedback, this system helps HR professionals rather than replacing them. It ensures that technology acts as a supportive tool in the job evaluation process.

References

Howan Anderson, Andreas Liujaya Wiranata, Henry Lucky, and Meiliana. 2024. Salary prediction with ensemble regressor model. In *2024 Ninth International Conference on Informatics and Computing (ICIC)*, pages 1–6.

Meenal Arora, Anshika Prakash, Amit Mittal, and Swati Singh. 2021. Hr analytics and artificial intelligence-transforming human resource management. In *2021 International Conference on Decision Aid Sciences and Application (DASA)*, pages 288–293.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. *Preprint*, arXiv:1706.03741.

David A. Cieslak and Nitesh V. Chawla. 2008. Learning decision trees for unbalanced data. In *Machine Learning and Knowledge Discovery in Databases*, pages 241–256, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yu Deng, Hang Lei, Xiaoyu Li, and Yiou Lin. 2018. An improved deep neural network model for job matching. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 106–112.

Sananda Dutta, Airiddha Halder, and Kousik Dasgupta. 2018. Design of a novel prediction engine for predicting suitable salary for a job. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 275–279.

Gemini Team. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, volume 35, pages 507–520. Curran Associates, Inc.

Mohammed Ayman Haseeb, Rishi Viswanathan, Kaushik Iyer, Anish Raj Hota, and Banu Priya Prathaban. 2024. Predictive salary modelling: Leveraging data science skills and machine learning for accurate forecasting. In *2024 9th International Conference on Communication and Electronics Systems (ICCES)*, pages 1011–1019.

Edward N. Hay. 1984. *The Hay Guide Chart-Profile Method of Position Evaluation*. McGraw-Hill, New York. Widely used point-factor system for job evaluation, focusing on internal factors.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Tarald O Kvålseth. 1985. Cautionary note about r^2. *The American Statistician*, 39(4):279–285.

Edward P. Lazear and Sherwin Rosen. 1981. Rank-order tournaments as optimum labor contracts. *Journal of Political Economy*, 89(5):841–864.

Antoine Magron, Anna Dai, Mike Zhang, Syrielle Montariol, and Antoine Bosselut. 2024. JobSkape: A framework for generating synthetic job postings to enhance skill matching. In *Proceedings of the First Workshop on Natural Language Processing for Human Resources (NLP4HR 2024)*, pages 43–58, St. Julian's, Malta. Association for Computational Linguistics.

Mayukh Maitra, Surabhi Sinha, and Tomas Kierszenowicz. 2024. An improved bert model for precise job title classification using job descriptions. In *2024 IEEE 12th International Conference on Intelligent Systems (IS)*, pages 1–6.

Yasser T. Matbouli and Suliman M. Alghamdi. 2022. Statistical machine learning regression models for salary prediction featuring economy wide activities and occupations. *Information*, 13(10).

M.L. Menéndez, J.A. Pardo, L. Pardo, and M.C. Pardo. 1997. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318.

Jacob Mincer. 1958. Investment in human capital and personal income distribution. *Journal of political economy*, 66(4):281–302.

Dena F. Mujtaba and Nihar R. Mahapatra. 2024. Fairness in ai-driven recruitment: Challenges, metrics, methods, and future directions. *arXiv preprint*.

Varsha Muralidharan, Bharathi Ravi, Sunita Pachar, Neelu Jain, Nilanjan Chakraborty, and D Sahaya Lenin. 2024. Ethical considerations in applying machine learning to hr practices: Balancing efficiency with fairness. In *2024 7th International Conference on Contemporary Computing and Informatics (IC3I)*, volume 7, pages 1392–1398.

European Parliament and Council of the European Union. 2024. Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence (artificial intelligence act). https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401689. Official Journal of the European Union, L 168, 1–169.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and 1 others. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Shakil Ahmed Pias, Meharaz Hossain, Hafizur Rahman, and Md. Mashrur Hossain. 2024. Enhancing job matching through natural language processing: A bert-based approach. In *2024 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 1–6.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Preprint*, arXiv:1908.10084.

Annalisa Wahyu Romadon, Kemas M Lhaksmana, Isman Kurniawan, and Donni Richasdy. 2020. Analyzing tf-idf and word embedding for implementing automation in job interview grading. In *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pages 1–4.

Sherwin Rosen. 1974. Hedonic prices and implicit markets: Product differentiation in pure competition. *Journal of Political Economy*, 82(1):34–55.

Patrick Schratz, Jannes Muenchow, Eugenia Iturritxa, Jakob Richter, and Alexander Brenning. 2019. Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecological Modelling*, 406:109–120.

Bruno Silva, Patricia Leite, and Óscar R. Ribeiro. 2025. Evaluating spam detection techniques: A comparison of tf-idf and sentence embeddings with machine learning models. In *2025 13th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–4.

Guanqi Wang. 2022. Employee salaries analysis and prediction with machine learning. In *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pages 373–378.

Azmine Toushik Wasi. 2024. HRGraph: Leveraging LLMs for HR data knowledge graphs with information propagation-based job recommendation. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 56–62, Bangkok, Thailand. Association for Computational Linguistics.

A Label distribution for Compas Grade and Salary

The *Grade* is a key feature representing job evaluation levels. As shown in Figure 2, most roles are between grades 9 and 14, while lower and higher grades have less frequency. This influences a potential imbalance in role evaluations because the dataset mainly contains white-collar jobs and does not include many blue-collar roles.

Figure 3 presents the distribution of the *Yearly Gross Salary* assigned to the job profiles. The salaries are not uniformly distributed. The distribution of Yearly Gross Salary is right-skewed, with most values are between €40,000 and €80,000.

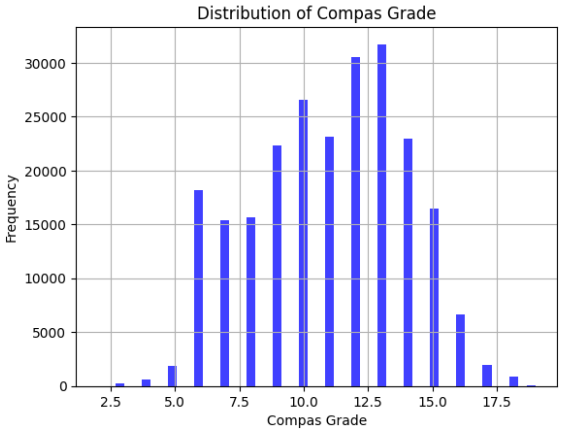

Figure 2: Distribution of Grade values

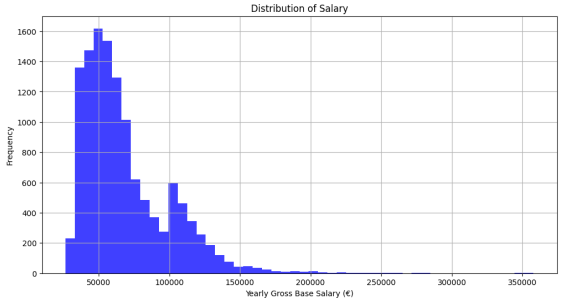

Figure 3: Distribution of Yearly Gross Salary

B Few-shot Examples Used in Prompting

The examples we used in our prompts are detailed in the following. Our aim is to create data in the format of our training set.

Example 1: Generate a structured feature string for the job title: `Senior Software Engineer`.

Domain (en): Technology; Specialisation (en): Backend Development; Career ladder (en): Expert; Level Title: Senior; HIERARCHIC MANAGEMENT: No; FUNCTIONAL MANAGEMENT: Yes, mentoring junior developers; Required experience: 5; Innovation: High – Contributes to new architecture; Responsible for a certain budget/figure?: No; Specify the budget amounts: N/A; Diploma: Bachelor's Degree; Speciality: Computer Science; Skill: Java/Spring; Skill: Cloud Platforms (AWS/Azure); Skill: CI/CD Pipelines; Process: Agile/Scrum; Internal Job Grade: 12; Department: Engineering;

Now, combine all these generated values, in the correct order, into a single line separated by `|`.

Generated Output: `Technology | Backend Development | Expert | Senior | No | Yes, mentoring junior developers | 5 | High - Contributes to new architecture | No | N/A | Bachelor's Degree | Computer Science | Skill: Java/Spring | Skill: Cloud Platforms (AWS/Azure) | Skill: CI/CD Pipelines | Process: Agile/Scrum | 12 | Engineering`

Example 2:

Job title: `Marketing Manager`.

First, think about and fill in the likely values for the following features for this job:

Domain (en): Sales & Marketing; Specialisation (en): Digital Campaign Management; Career ladder (en): Manager; Level Title: Manager; Hierarchic Management: Yes, manages a team of 3; Functional Management: Yes, leads campaign strategy; Required experience: 4; Innovation: Medium – Develops new campaign ideas; Responsible for a certain budget/figure?: Yes; Diploma: Master's Degree; Speciality: Marketing or Business Administration; Tool: Google Analytics; Tool: Salesforce Marketing Cloud; Skill: SEO/SEM Strategy; Skill: Content Marketing; Internal Job Grade: 11; Department: Marketing.

Generated Output: `Sales & Marketing | Digital Campaign Management | Manager | Manager | Yes, manages a team of 3 | Yes | 4 | Medium - Develops new campaign ideas | Yes | Master's Degree | Marketing or Business Administration | Tool: Google Analytics | Tool: Salesforce Marketing Cloud | Skill: SEO/SEM Strategy | Skill: Content Marketing | 11 | Marketing`

C Model training configuration

Tables 5, 6, 7 show the training configurations for the models.

Hyperparameter	Value
Task	Singlelabel multiclass classification (321 classes)
Optimizer	AdamW
Training Loss	Cross-entropy
Learning Rate (lr)	1×10^{-5}
Number of Epochs	3
Batch Size	8
Random Seed	42
Pre-trained Model	RoBERTa-base
Token Max Length	512
Loss Averaging	Weighted or Macro
Weight Decay	0.01

Table 5: Model training configuration for job title classification with RoBERT

Hyperparameter	Value
Task	Regression (Grade & Salary prediction)
Transformer embeddings	all-MiniLM-L6-v2 / all-mpnet-base-v2 / paraphrase-MiniLM-L6-v2
Embedding dimensions	384 / 768 / 384
Train/Test Split	test_size=0.20, random_state=11
Hidden Layers	[512, 256, 256, 256]
Activation Function	ReLU
Dropout Rate	0.20
Optimizer	Adam
Learning Rate	1×10^{-3}
Loss Function	Mean Squared Error (MSE)
Batch Size	32
Number of Epochs	40

Table 6: Model training configuration for Grade & Salary regression.

D Grid Search

We perform a grid search with 3-fold cross-validation to tune the RandomForestRegressor and XGBoost regressor hyperparameters with scikit-learn GridSearchCV (Schratz et al., 2019) The best-performing hyperparameter combination, as determined by the lowest cross-validation error. The results of this procedure are in Table 8.

E Job evaluator application

Figures 4 and 5 show screenshots of the job evaluator application implemented.

Hyperparameter	Value
Task	Regression (Grade & Salary prediction)
Pre-trained Model	distilroberta-base
Loss Function	MSELoss
Optimizer	AdamW
Learning Rate	2×10^{-5}
Dropout	0.2
Max Sequence Length	64
Tokenization Batch Size	16
Batch Size (training)	8
Number of Epochs	4
Random Seed	42

Table 7: Training configuration and model settings for DistilRoBERTa-based regression on Compas Grade

Model	Hyperparameter	Description	Search Range	Best Value
XGBoost	`n_estimators`	Number of boosting rounds (trees)	{100, 500}	500
	`max_depth`	Maximum tree depth per estimator	{4, 6, 8, None}	6
	`learning_rate`	Controls how fast we learn	{0.01, 0.05, 0.1}	0.05
	`subsample`	Fraction of samples used per tree	{0.6, 0.8, 1.0}	0.8
Random Forest	`n_estimators`	Number of trees in the ensemble	{100, 200}	200
	`max_depth`	Maximum depth of each tree	{None, 10, 20}	None
	`min_samples_split`	Minimum number of samples to split	{2, 10}	2
	`min_samples_leaf`	Minimum number of samples at leaf	{1, 4}	1
	`max_features`	Number of features per split	{`"sqrt"`, 0.5}	0.5

Table 8: Grid search ranges and best-found values for XGBoost and RandomForestRegressor hyperparameters on the Grade & Salary prediction task.

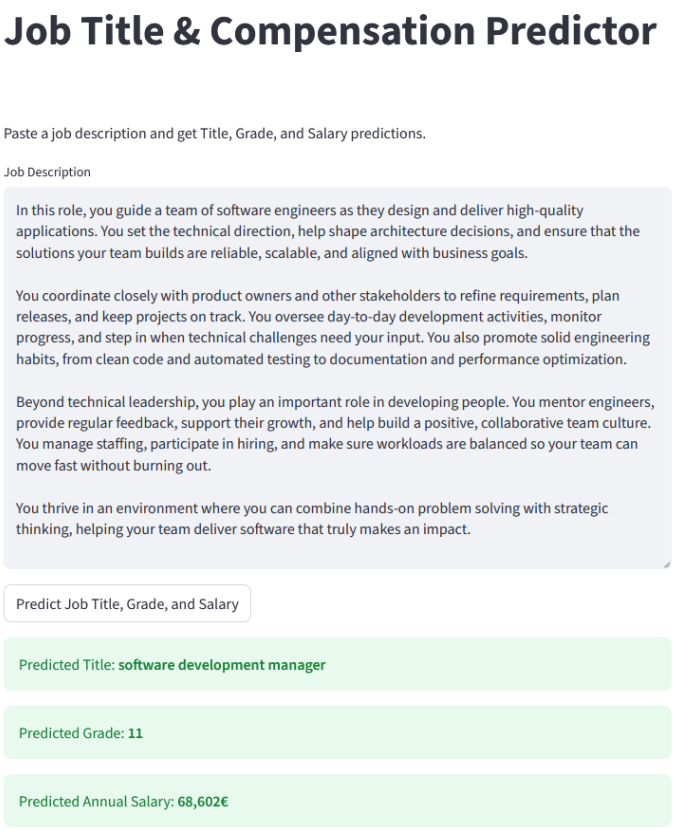

Figure 4: Output of the job evaluator showing the predicted job title, grade, and annual salary.

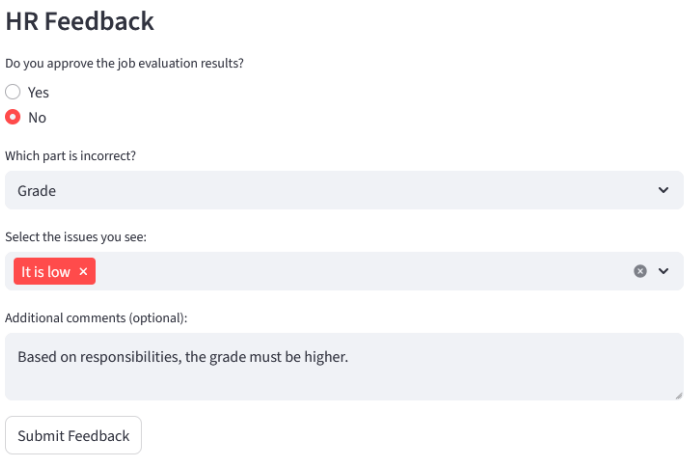

Figure 5: HR feedback saved in a separate file for future model improvement.