aiXplain SDK: A High-Level and Standardized Toolkit for AI Assets

Shreyas Sharma[*] **Lucas Pavanelli**[*] **Thiago Castro Ferreira**
Mohamed Al-Badrashiny and **Hassan Sawaf**
aiXplain Inc.,
California, USA
{shreyas,lucas.pavanelli,thiago,mohamed,hassan}@aixplain.com

Abstract

The aiXplain SDK[1] is an open-source Python toolkit which aims to simplify the wide and complex ecosystem of AI resources. The toolkit enables access to a wide selection of AI assets, including datasets, models, and metrics, from both academic and commercial sources, which can be selected, executed and evaluated in one place through different services in a standardized format with consistent documentation provided. The study showcases the potential of the proposed toolkit with different code examples and by using it on a user journey where state-of-the-art Large Language Models are fine-tuned on instruction prompt datasets, outperforming their base versions.

1 Introduction

A software development kit (SDK) is a collection of software development tools in one installable package (Wikipedia contributors, 2024). The popularity of these toolkits in AI stems from their powerful features, ease of use, and applications in diverse fields including deep learning (Pedregosa et al., 2011; Abadi et al., 2015; Paszke et al., 2019), computer vision (Itseez, 2015), natural language processing (Bird et al., 2009; Manning et al., 2014; Qi et al., 2020), and beyond. This wide range of options available, however, can make it difficult to combine services from different SDKs into one application, since the integration requires a deep understanding of the usage, dependencies, and intricacies of each technology.

To address this challenge, we introduce the aiXplain SDK, a unified platform providing seamless access to a diverse collection of AI resources, including datasets, models, and metrics. By integrating both open-source and commercial options, this SDK abstracts complexities such as hosting and billing, streamlining the research process. The SDK's flexibility that allows for effortless swapping of components by just changing the asset id enables faster prototyping. Furthermore, the standardization of metrics and datasets within the SDK creates a level playing field for comparative analysis by mitigating the influence of disparate evaluation methodologies. Researchers can efficiently discover, utilize, and assess these resources in a standardized, well-documented environment.

```python
from aixplain.factories import (
    ModelFactory
)
model = ModelFactory.get(
    "60ddefa08d38c51c5885e760"
)
response = model.run("Hello, World!")
```

Figure 1: Model Execution example on the SDK

The aiXplain SDK aims to help both Artificial Intelligence users and developers. Figure 1 exemplifies how with a few lines of code users can embed a Machine Learning model from the aiXplain marketplace into their application. For developers, the proposed SDK covers the entire Machine Learning development lifecycle, allowing them to select/onboard data as well as to train, evaluate and serve their models.

The SDK's Python code is released under the Apache-2.0 license and is publicly accessible on GitHub[1], where comprehensive documentation and tutorials are also available. The getting started guide[*], along with the tutorial series[1] is prepared to help new users get familiar with the toolkit. A Demo[1] is also provided to see the capabilities of SDK in action for a real world use-case. This setup helps new users to get started quickly and facilitates easy contributions from the entire community to the project.

[*]These authors contributed equally to this work
[1]GitHub: https://github.com/aixplain/aiXplain
Demo: https://youtu.be/WZVuh99gJDg
Series: https://www.youtube.com/playlist?list=PL4X2zpOPPGeq2lbzmfn04aCPNqimalhQJ

[*]https://github.com/aixplain/aiXplain/blob/main/docs/development/developer_guide.md

2 Modules

Figure 2 depicts the architecture of the proposed SDK. The toolkit was designed to handle different kinds of assets such as Corpora, Datasets, Models and Metrics. In this section, we delve into each of these core modules, detailing their functionalities and highlighting how they converge to enhance overall performance and streamline user interactions within the system.

2.1 Corpus and Dataset

In the SDK toolkit, we differentiate data assets between "corpora" and "datasets". A corpus is designed as a flexible, context-rich collection of data, intended for general and exploratory data analysis use cases. On the other hand, a dataset consists of a compilation of data with specified inputs and outputs focused on a specific ML task (e.g. Speech Recognition, Machine Translation, Sentiment Analysis, etc). Datasets are tailored for specific research questions or applications that require fine-tuning or benchmarking an ML model. As an example of usage, Figure 3 depicts how to list English Speech Synthesis datasets available in the aiXplain marketplace using the SDK.

2.2 Model

The proposed SDK serves as a gateway to a curated selection of machine learning models from diverse commercial suppliers and the AI community at large, precisely matching users with the models that align with their specific needs. This is achieved through an organized catalog that classifies models based on functionality, input/output type, and supplier among other criteria. . The platform currently hosts a comprehensive collection of over 40,000 models across 30+ AI applications, with the repository expanding at a rapid pace. Figure 4 exemplifies how to list text generation models in the aiXplain marketplace.

2.3 Metric

The SDK places a significant emphasis on the evaluation phase of AI models by providing a wide-range of evaluation metrics. For Text Generation tasks, it includes classical metrics such as BLEU (Papineni et al., 2002) and WER (Woodard and Nelson) but also expands to encompass state-of-the-art metrics trained with human evaluation scores like Comet DA (Rei et al., 2020), and reference-less ones such as Nisqa (Mittag et al., 2021). Our toolkit supports 30+ metrics, covering a wide variety of tasks and modalities. It includes built-in metrics designed for evaluating the performance of specific AI tasks like Machine Translation (e.g., TER (Snover et al., 2006), METEOR Banerjee and Lavie, 2005), Speech Recognition (e.g., WIL, MER (Morris et al., 2004)), and Speech Synthesis (e.g., PESQ (Rix et al., 2001), DNSMOS (Reddy et al., 2021)). Figure 5 shows how to run the BLEU metric.

3 Services

Inherent to the Machine Learning (ML) lifecycle, it is crucial to consider the multifaceted roles and needs of AI professionals who contribute to the successful development, deployment, and maintenance of ML models. As depicted in Figure 2, the design of the proposed SDK centers on forging a unified and collaborative ecosystem tailored for the wide spectrum of AI professionals engaging in the ML development lifecycle. In the following subsections we explain in detail each of these services.

3.1 Data Asset Onboard

Figure 7 depicts an example of use of the Dataset Onboard service of the SDK, where a demo data-to-text dataset is onboarded. A new data asset is onboard in the aiXplain marketplace from a CSV file where each column represents a data.

3.2 FineTune

The FineTune service aims to help Data Scientists fine-tune a model for a specific task using a collection of focused datasets. Figure 8 depicts a template for coding the process in the SDK. During the training process, the user can check information about the training procedure status (line 14), which shares relevant metrics, such as train and evaluation losses, epoch, and learning rate. Once the process is done, the finetuned model is served for inference as any other model, making easy the work of ML Engineers.

3.3 Benchmark

The Benchmark service in our SDK toolkit sets a new standard in evaluating AI models, providing a seamless and in-depth analysis across various tasks and domains. Designed with a strong emphasis on modularity and interoperability, it utilizes our extensive array of existing modules - models, datasets, and metrics.

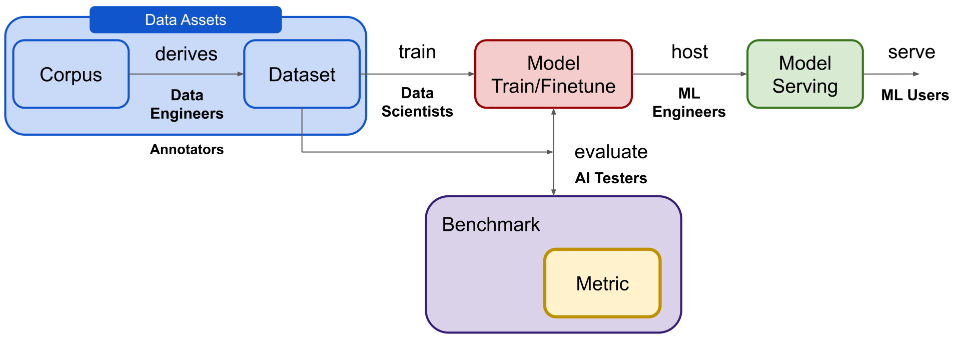

Figure 2: System Architecture of the proposed SDK

```python
from aixplain.factories import (
    DatasetFactory
)
from aixplain.enums import (
    Function,
    Language
)
datasets = DatasetFactory.list(
    function=Function.SPEECH_SYNTHESIS,
    source_languages=Language.ENGLISH
)
```

Figure 3: Listing English Speech Synthesis datasets on the SDK

```python
from aixplain.factories import (
    ModelFactory
)
from aixplain.enums import (
    Function,
    Language
)
models = ModelFactory.list(
 function=Function.TEXT_GENERATION,
)
```

Figure 4: Listing Text Generation models on the SDK

```python
from aixplain.factories import (
    MetricFactory
)
bleu_metric = MetricFactory.get(
    "639874ab506c987b1ae1acc6"
)
response = bleu_metric.run(
    hypothesis=[
        "sample hypothesis 1",
        "sample hypothesis 2"
    ],
    reference=[
        "sample reference 1",
        "sample reference 2"
    ]
)
```

Figure 5: Metric Execution example on the SDK

```python
from aixplain.factories import (
    BenchmarkFactory,
    DatasetFactory,
    MetricFactory,
    ModelFactory
)

datasets = DatasetFactory.list("...")
metrics = MetricFactory.list("...")
models = ModelFactory.list("...")

benchmark = BenchmarkFactory.create(
    "benchmark_name",
    dataset_list=datasets,
    model_list=models,
    metric_list=metrics
)
job = benchmark.start()
status = job.check_status()
results = job.download_results_as_csv()
```

Figure 6: Benchmark example on the SDK

This service goes beyond traditional leaderboards by offering a nuanced analysis including model performance, latency, and operational cost, ensuring a holistic and in-depth comparison of models. Moreover, we incorporated a cutting-edge, LLM-powered interpreter that offers users, regardless of their expertise level, lucid explanations of their benchmarking outcomes, enhancing understanding and facilitating informed decision-making. Additionally, it incorporates a bias analysis feature, ensuring any detected biases are highlighted so that they can be addressed, underscoring the commitment to fairness and ethical AI development. Figure 6 depicts the template for setting a benchmark job in the SDK.

4 User Journey

This section presents a complete user journey, walking through all SDK's modules and services, demonstrating how to (1) Onboard train and test

Model name	Baseline	Fine-tuned
Llama 2 7b	0.71	0.74
Mistral 7b	0.76	0.76
Solar 10.7b	0.53	0.72

Table 1: Evaluation of baseline and fine-tuned models on PubMedQA dataset.

datasets, (2) Fine-tune LLMs on train datasets and (3) Benchmark baseline and fine-tuned LLMs on test datasets.

4.1 Onboarding datasets

We selected and onboarded into the aiXplain platform the following well-known open-source datasets:

PubMedQA (Jin et al., 2019) is a biomedical question-answering (yes/no/maybe) dataset collected from PubMed abstracts. **Alpaca** (Taori et al., 2023) consisting of 52k instruction-following data. It was used to train our LLMs to follow instructions. **Truthful QA** (Lin et al., 2022) is a dataset consisting of multiple choice questions. We used it as an evaluation task with two defined scores: MC1, in which the model must select a single answer out of the choices, and MC2, the model can select multiple correct answers.

4.2 Fine-tuning LLMs

For fine-tuning, we selected three baseline models from the aiXplain marketplace:

Llama 2 7b (Touvron et al., 2023) from Meta, **Mistral 7b** (Jiang et al., 2023) by Mistral AI and **Solar 10.7b** (Kim et al., 2023) by Upstage AI.

We fine-tuned all three models on the PubMedQA train set and the entire Alpaca dataset for one epoch, using 1e-5 as the learning rate and gradient checkpointing. We also utilized the LoRA (Hu et al., 2021) method to save memory when fine-tuning the LLMs.

4.3 Benchmarking

In our user journey, we conducted Benchmarks to evaluate the performance of the above LLMs on multiple choice tasks. We used accuracy as the main metric and compared the generated loglikelihoods of the possible choices.

For the models trained on the PubMedQA train set, we evaluated them on the PubMedQA test set, testing whether the models' capabilities are adequate for the biomedical domain. Secondly, for the models trained on the Alpaca dataset, we benchmarked them on the Truthful QA dataset, which measures the LLMs' ability to follow general knowledge instructions.

	Truthful MC1		Truthful MC2	
Model name	**B**	**Ft**	**B**	**Ft**
Llama 2 7b	0.25	0.38	0.39	0.54
Mistral 7b	0.28	0.38	0.43	0.54
Solar 10.7b	0.58	0.44	0.72	0.61

Table 2: Evaluation of baseline and fine-tuned models on Alpaca dataset. **B** refers to baseline models and **Ft** to fine-tuned ones.

4.4 Results and Discussion

Table 1 shows the results for PubMedQA dataset. For all LLMs, fine-tuned models outperformed baseline ones. These results show that primarily Solar 10.7b benefits greatly from the training process, with fine-tuned LLM improving 37% in accuracy over the baseline.

Table 2 shows the results for models fine-tuned on the Alpaca dataset. For Llama 2 7b and Mistral 7b, the training process dramatically improves the model for the Truthful QA task, improving Llama 2 7b 39% for Truthful MC2 task. However, for Solar 10.7b, fine-tuning does not enhance the performance, which may be attributed to the already excellent baseline model performance.

It is also worth pointing out that the development time using the SDK is much less than other options. We used less than 20 lines of code to conduct the whole user journey and did not need to set up any other Python packages or hardware infrastructure. For example, considering the fine-tuning LLM step, we used only 8 lines, as depicted in Figure 8, without the need to own any hardware. On the other hand, HuggingFace's Transformers uses approximately 150 lines and requires the allocation of more expensive GPUs.

5 Related Work

Scikit Learn (Pedregosa et al., 2011) is an example of a traditional Machine Learning SDK. The toolkit is known by its simplicity and accessibility to apply traditional Machine Learning algorithms for problems that involve structured data.

PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2015) are examples of more recent SDKs used in the development of state-of-the-art deep learning models. On top of them, other high-level frameworks were proposed such as Hugging-Face's Transformers (Wolf et al., 2020) and Keras (Chollet et al., 2015).

Software development kits have also been proposed for specific Machine Learning tasks such as OpenCV (Itseez, 2015) for Computer Vision; and NLTK (Bird et al., 2009), Stanford CoreNLP (Manning et al., 2014) and Stanza (Qi et al., 2020) for Natural Language Processing.

Popular cloud services also make their own SDKs available to manipulate their services programmatically, including the AI ones. This is the case for Google[*] and AWS[*] cloud services.

Within this wide and complex ecosystem, the SDK aims to be a marketplace where the AI assets and tools provided by other suppliers and SDK could be found into a single, standardized and well-documented access point.

6 Conclusion

This study demonstrates how complicated can be the creation of an AI application combining assets from the wide and complex range of software toolkits in the field. To solve this problem, we propose the aiXplain SDK which enables access to AI corpora, datasets, models and metrics from different commercial and community sources in a standardized format. Through straightforward, well-documented, and exemplified services, the toolkit enables onboarding data assets as well as finetuning, evaluating, serving, and using AI models. The toolkit's potential is demonstrated in a user journey where three state-of-the-art large language models are fine-tuned on instruction prompt question-answering datasets. After the fine-tuning process, an evaluation is conducted in the proposed SDK demonstrating how the trained models outperformed the base ones.

Finally, the toolkit is publicly available on Github and released under an open-source license (Apache-2.0) along with a demo, example notebooks and video tutorials. We hope the community engages in its use and development, contributing to its growth.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

François Chollet et al. 2015. Keras. `https://keras.io`.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Itseez. 2015. Open source computer vision library. `https://github.com/itseez/opencv`.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. 2023. Solar 10.7b: Scaling large language models with simple yet effective depth up-scaling.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

[*] `https://cloud.google.com/sdk`

[*] `https://aws.amazon.com/developer/tools`

Gabriel Mittag, Babak Naderi, Assmaa Chehadi, and Sebastian Möller. 2021. Nisqa: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets. *arXiv preprint arXiv:2104.09494*.

Andrew Cameron Morris, Viktoria Maier, and Phil Green. 2004. From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In *Proc. Interspeech 2004*, pages 2765–2768.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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. *Advances in neural information processing systems*, 32.

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.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Chandan K A Reddy, Vishak Gopal, and Ross Cutler. 2021. Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6493–6497.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Unbabel's participation in the WMT20 metrics shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 911–920, Online. Association for Computational Linguistics.

A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. 2001. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 2, pages 749–752 vol.2.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Wikipedia contributors. 2024. Software development kit. [Online; accessed 17-03-2024].

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.

J.P. Woodard and year = 1982 journal = Workshop on standardisation for speech I/O technology, Naval Air Development Center, Warminster, PA title = An information theoretic measure of speech recognition performance Nelson, J.T.

A Code Snippets

Supplementary code snippets for the SDK's various modules and services.

```python
import pandas as pd
from aixplain.factories import DatasetFactory
from aixplain.modules import MetaData
from aixplain.enums import Function, Language, License

df = pd.DataFrame({
    "data": [
        "Joe_Biden president United_States",
        "South_Africa capital Cape_Town"
    ],
    "en": [
        "Joe Biden is the president of the United States.",
        "The capital of South Africa is Cape Town."
    ]
})
df.to_csv("dataset.csv")

data_meta = MetaData(
    name="data",
    dtype="text",
    storage_type="text",
)

en_meta = MetaData(
    name="en",
    dtype="text",
    storage_type="text",
    languages=[Language.English]
)

payload = DatasetFactory.create(
  name="dataset_demo",
  description="Data2Text Dataset",
  license=License.MIT,
  function=Function.TEXT_GENERATION,
  content_path="dataset.csv",
  input_schema=[data_meta],
  output_schema=[en_meta]
)
```

Figure 7: Dataset Onboard example on the SDK

```python
from aixplain.factories import DatasetFactory, ModelFactory, FinetuneFactory

dataset = DatasetFactory.get("...")
model = ModelFactory.get("...")
finetune = FinetuneFactory.create(
    "finetuned_model",
    [dataset],
    model
)
finetuned_model = finetune.start()
finetuned_model.check_finetune_status()
```

Figure 8: Model Fine-tuning example on the SDK