رواد مصر الرقمية

Amazon Customer Product Reviews

Table Of Contents

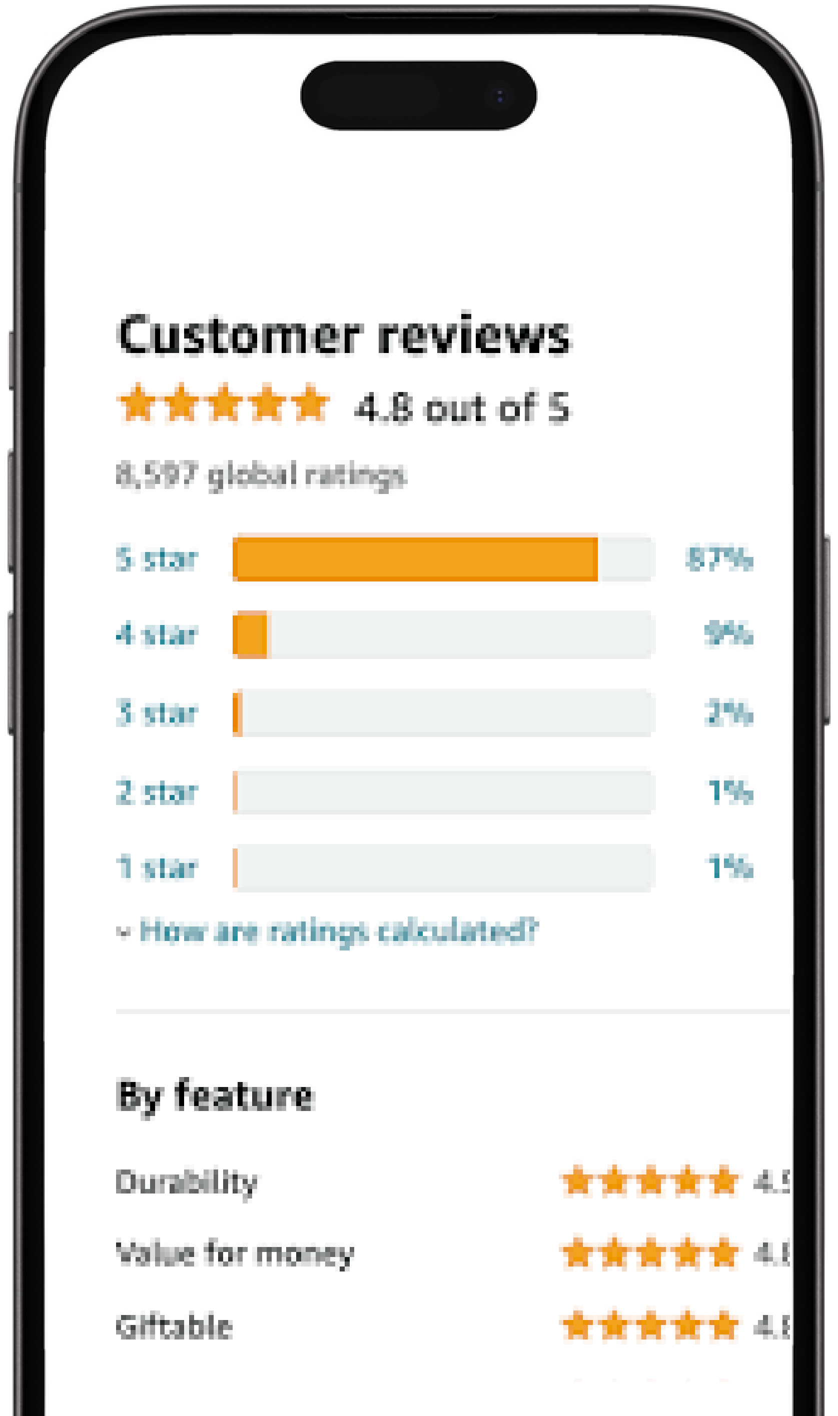
Introduction

We performed sentiment analysis on Amazon customer reviews to classify them as positive or negative. The project involves using machine learning and deep learning models, including LSTM, to achieve high accuracy in predicting customer sentiment. Additionally, we implemented platforms like Hugging Face, MLflow, and Streamlit for model optimization, tracking, and deployment.

Team Members

- Mariam Mohamed
- Haidy Talaat
- Ahmed Emad
- Mostafa Omran
- Abdelrahman Tarek
- Abdelrahman Elsheikh
- Fatma Salah

DATASET



Customer reviews
4.8 out of 5
8,597 global ratings
5 star 87%
4 star 9%
3 star 2%
2 star 1%
1 star 1%
How are ratings calculated?
By feature
Durability
Value for money
Giftable

Dataset **Overview**

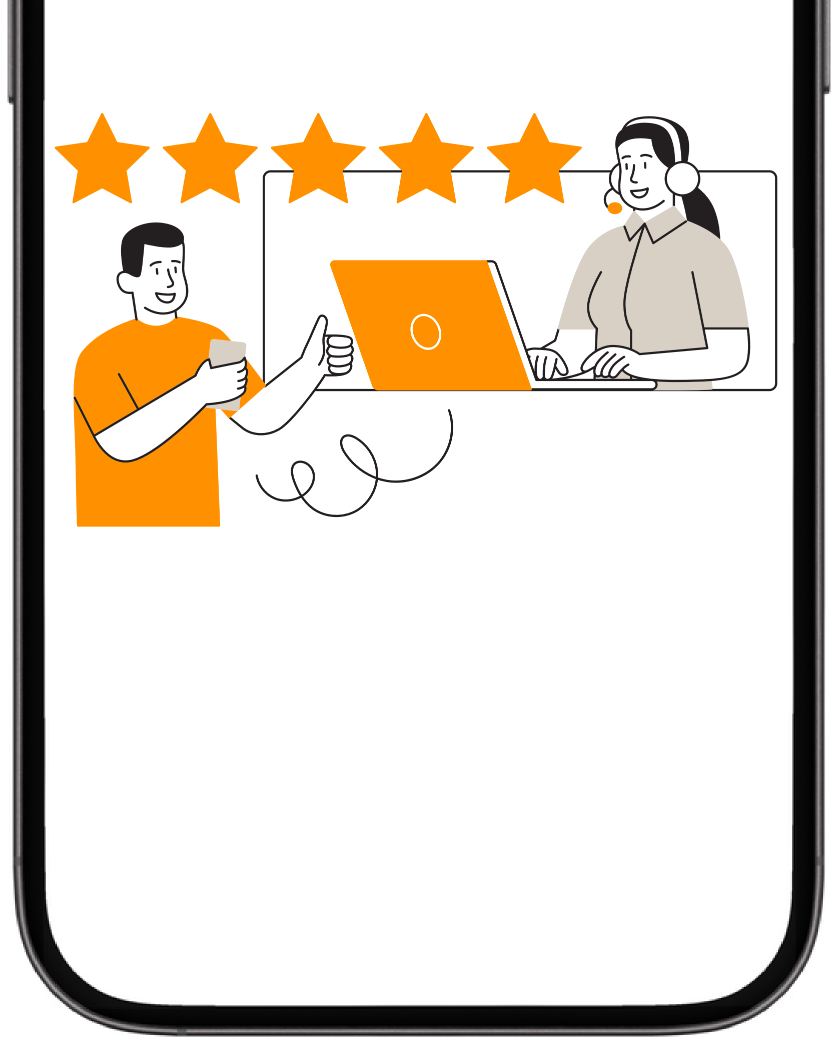
This dataset contains over 568,000 consumer reviews ratings scored 1-5 stars for a variety of Amazon products.

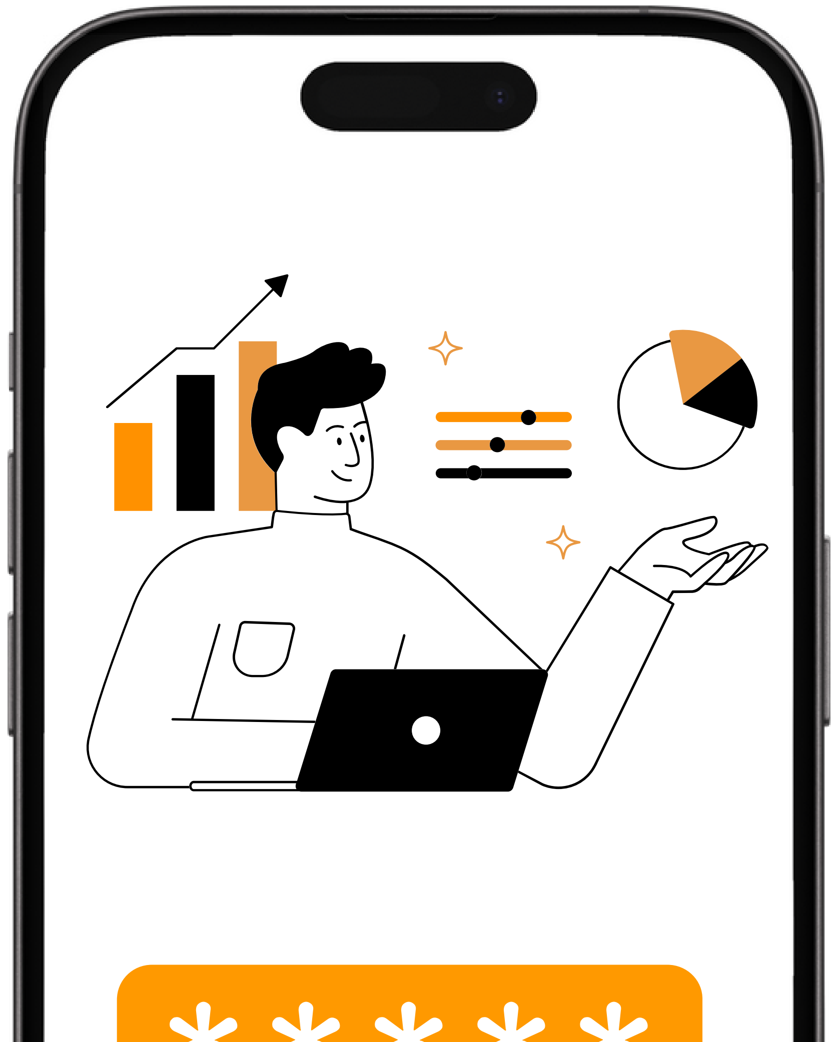
Data dictionary:

- **Id:** Unique identifier for each review
- **ProductId:** Amazon product ID
- **UserId:** Amazon user ID
- **ProfileName:** User's profile name
- **HelpfulnessNumerator:** Number of users who found the review helpful
- **HelpfulnessDenominator:** Total number of votes for the review
- **Score:** Review rating (1-5 stars)
- **Time:** Timestamp of the review
- **Summary:** Short summary of the review
- **Text:** Full review text

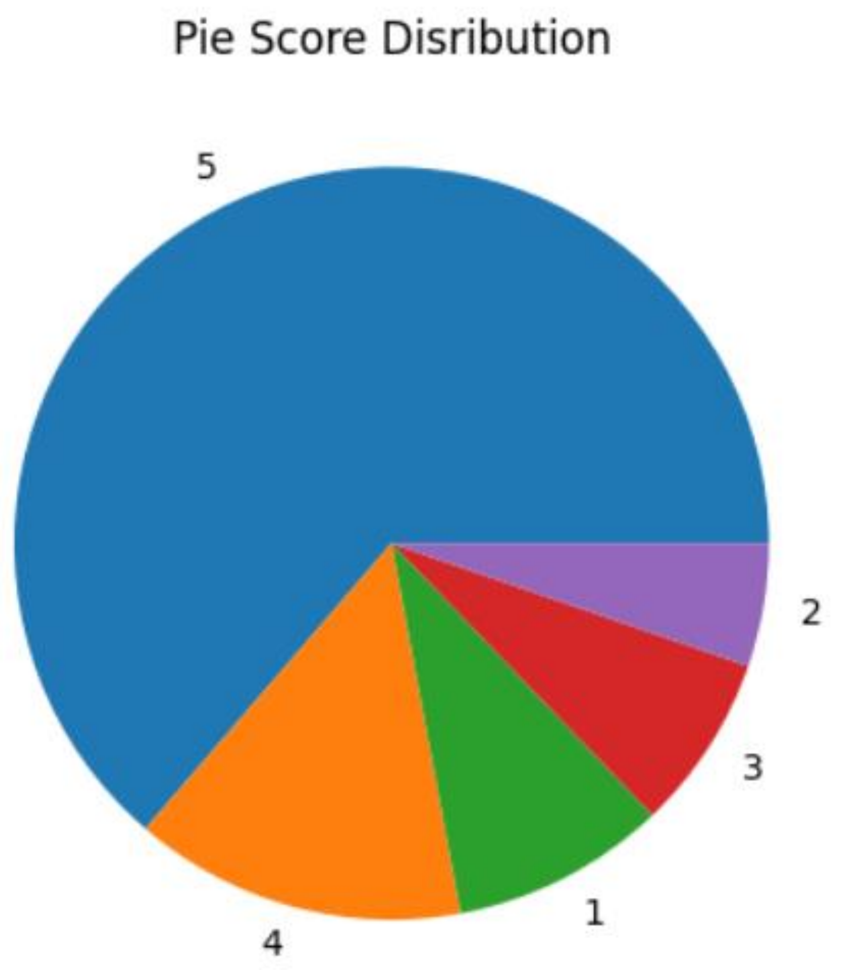
EXPLORATORY DATA ANALYSIS (EDA)

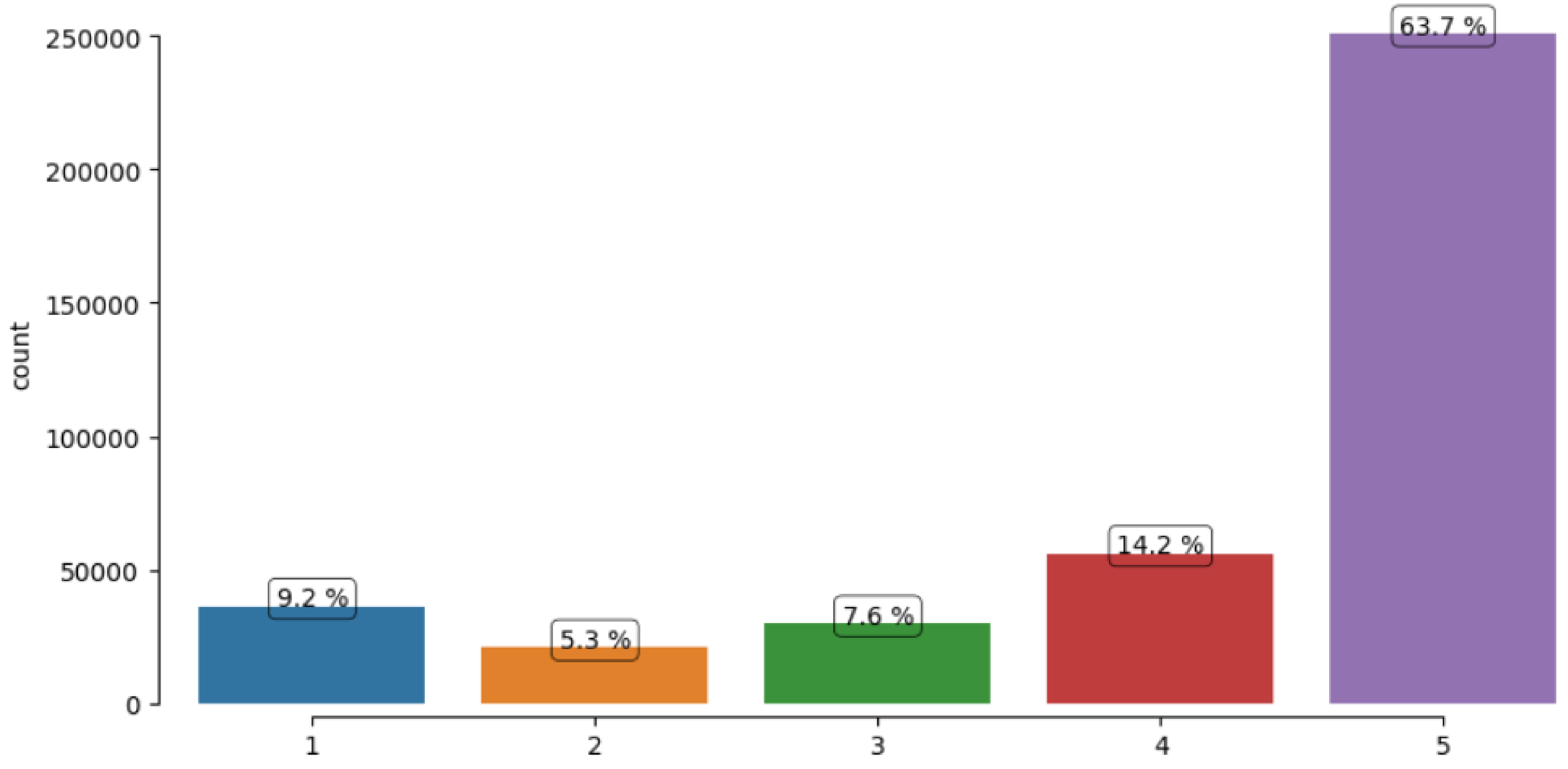
1. Score distribution
2. Word Cloud
3. Data Duplication





Score **Distribution**

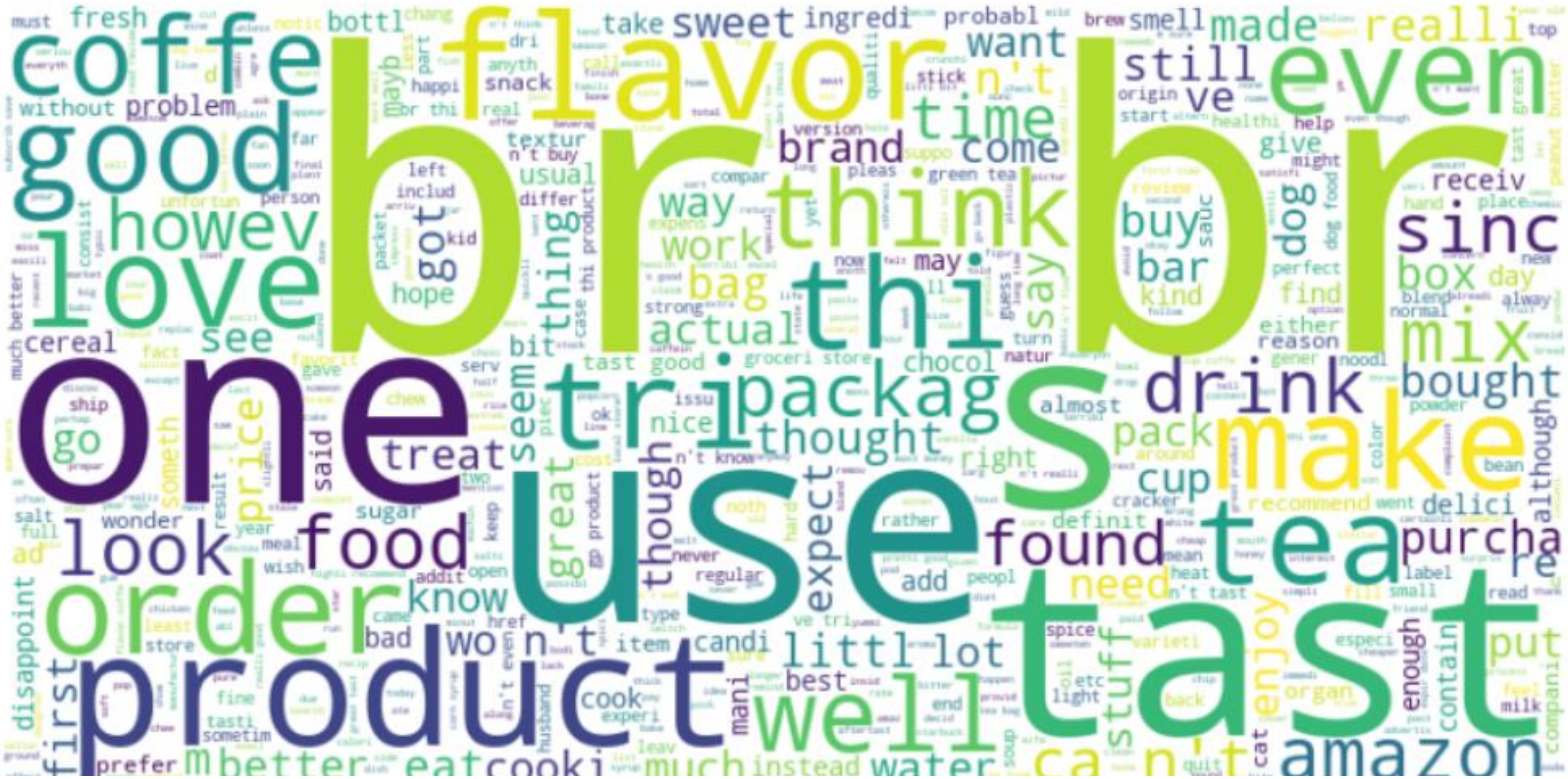




Key Observations:

- **Imbalanced data**
- **Fewer lower scores:** Ratings of 1, 2, and 3 are relatively infrequent, indicating a lower prevalence of negative or neutral sentiment.

Word Cloud



Key observation:

- **The word cloud visually represents the most frequently used terms in the dataset, highlighting words like "coffee," "tea," "taste," "good," "product," and "use,"** suggesting that these topics are central to the customer reviews.

Data Duplication

```
# To check for duplicates
df.duplicated().sum()
```

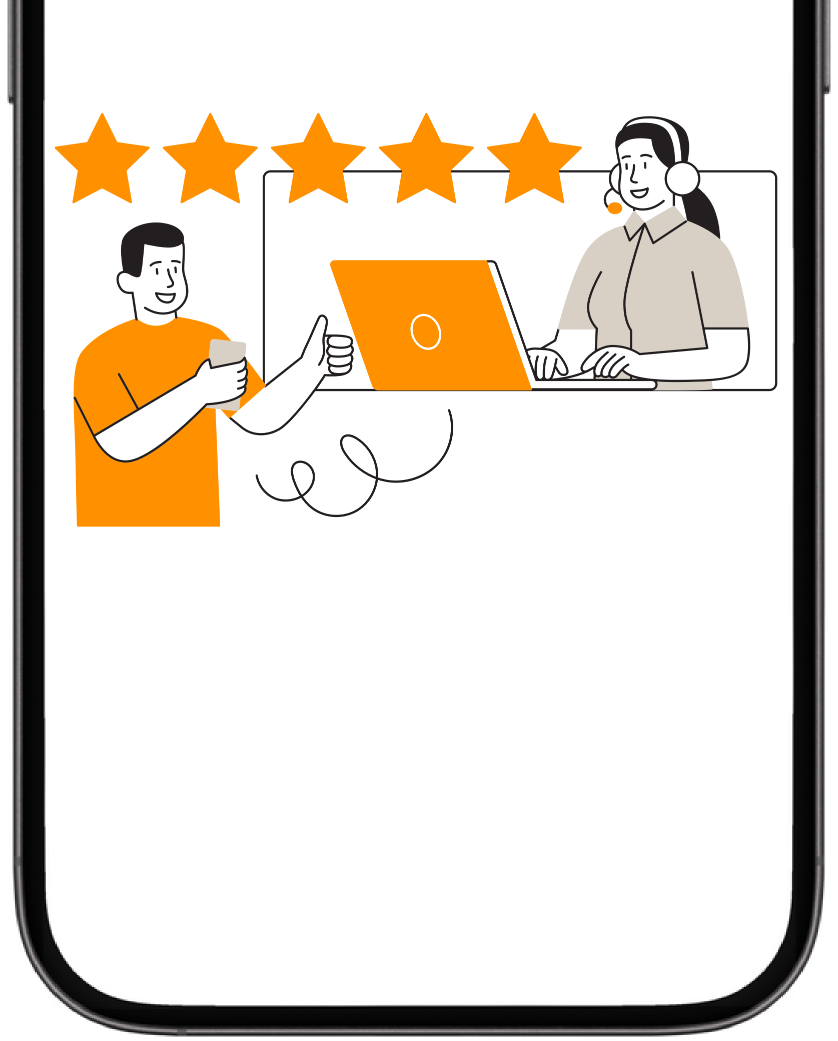
```
174779
```

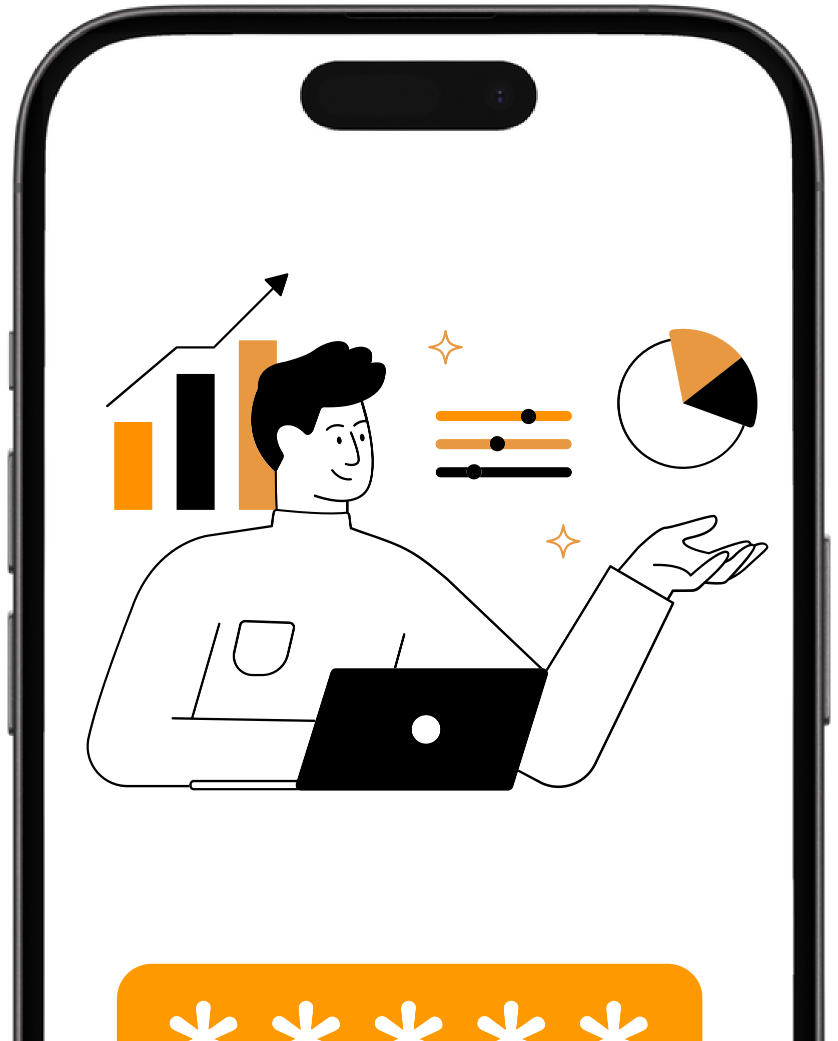
Key observation:

- **There are 174779 duplicated reviews.**

PREPROCESSING AND FEATURE ENGINEERING

1. Dropping Duplicated Values
2. Sampling Data
3. Text Preprocessing
4. Score Encoding





Dropping Duplicated values

```
df.drop_duplicates(inplace=True)
```

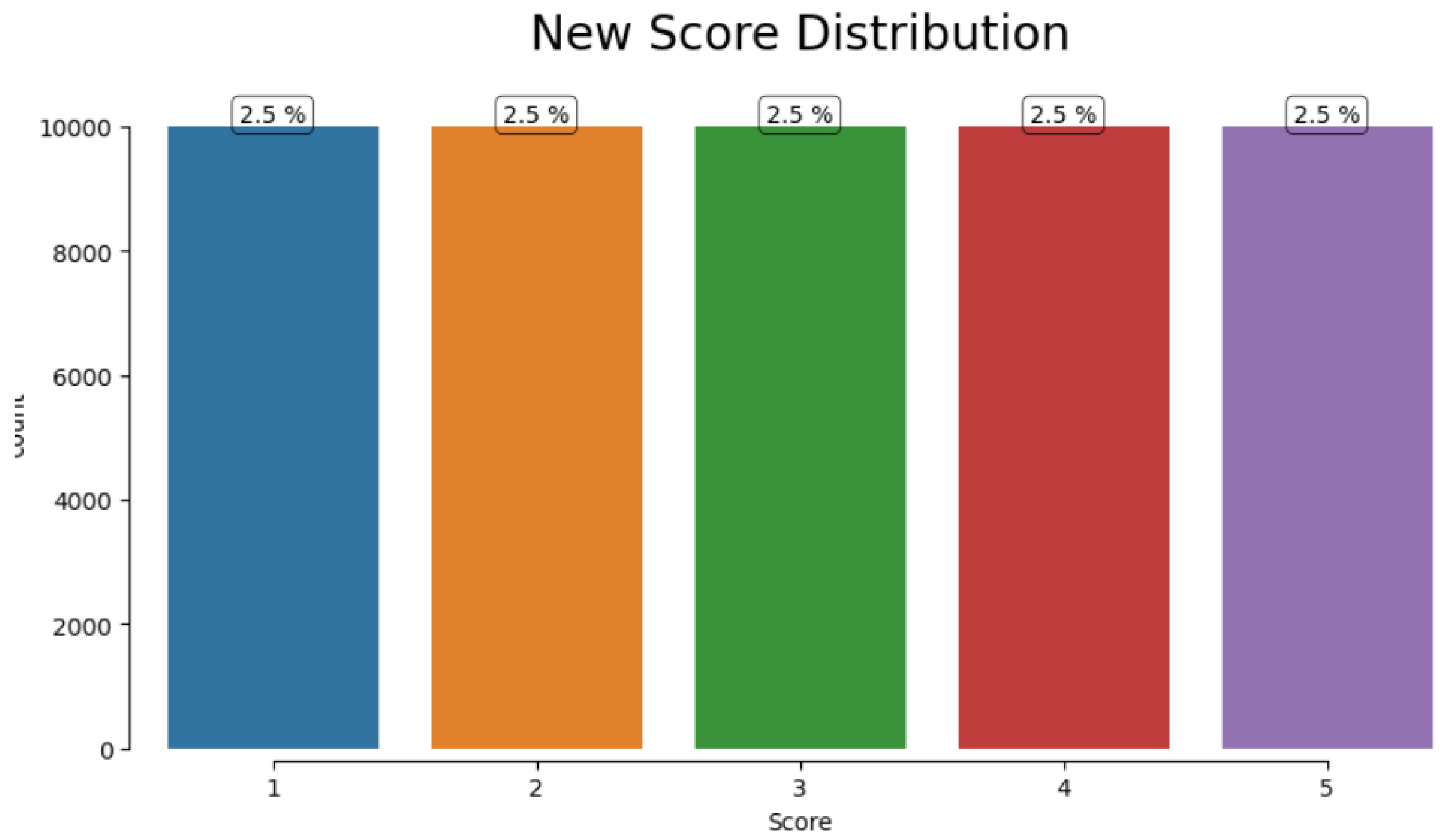
```
df.shape
```

```
(393675, 2)
```

To ensure the integrity of our dataset, we removed duplicate entries. The original dataset had 568454 and after dropping 174779 duplicates, the new dataset dimension is 393675. This step helps eliminate redundant data, improving the quality of the analysis.

As we don't need the other attributes, we drop all except the text and score.

Sampling Data



We implemented a balanced sampling strategy to address class imbalance by randomly selecting 10,000 reviews from each score category. This approach ensures equal representation of all score categories, facilitating more accurate sentiment analysis.

Text Preprocessing

```
def clean_text(text):
    # 1. Convert to lower
    txt=text.lower()

    # 1. split to words
    tokens=word_tokenize(text)

    # 3. remove punctuation
    tokens=[word for word in tokens if word not in string.punctuation]

    # 4. Remove stopwords
    tokens=[word for word in tokens if word not in stop_words]

    # 5. Remove numbers
    tokens=[word for word in tokens if not word.isdigit()]

    # 6. Apply Stemming
    tokens=[stemming.stem(word) for word in tokens]

    # To return these single words back into one string
    return ' '.join(tokens)
```

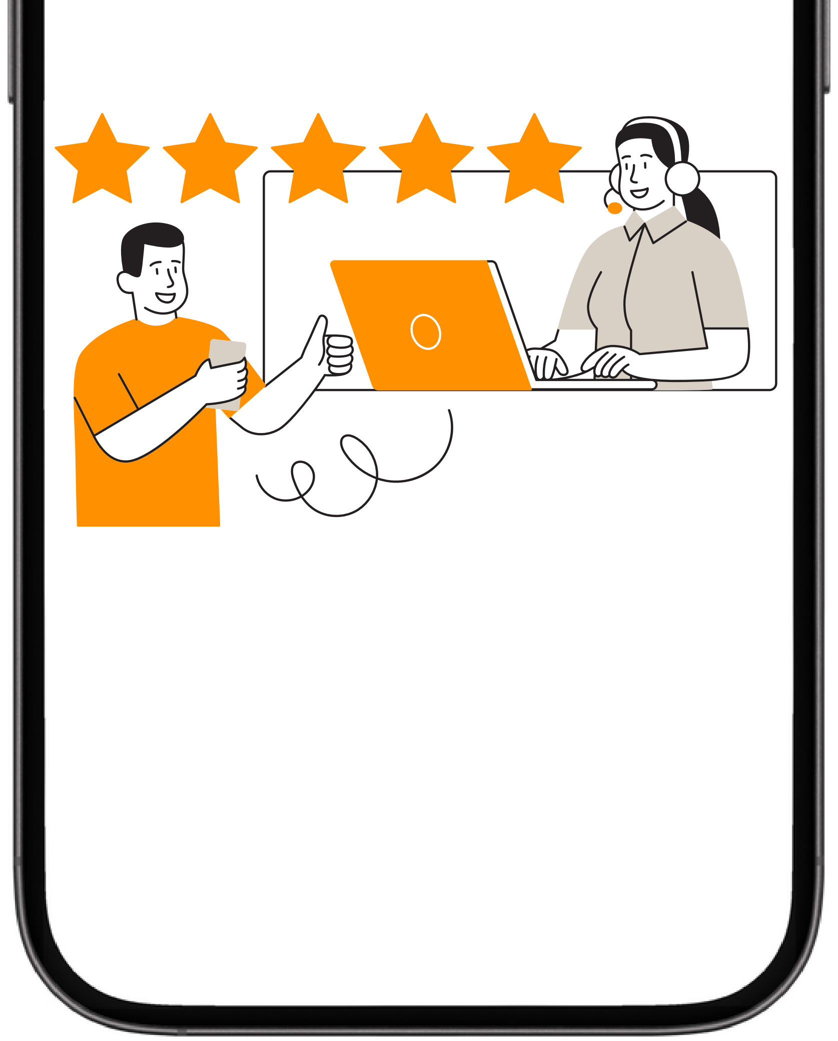
Text	cleaned_text
The wheat free brownie mix is not to my liking...	the wheat free browni mix like I wo n't reorde...
saltyness may be the "norm" for products like ...	salty may `` norm '' product like doesnt mean ...
I purchased this product from Otto's because t...	I purchas product otto 's offer via amazon pri...
Switch to this food and my dog became very sic...	switch food dog becam sick Go onlin lookup dog...
please do not take this note as an attack on y...	pleas take note attack product It snoodl hate ...
...	...
I bought this for Halloween and I had SOOOOOO ...	I bought halloween I soooooo much candi left I...
My two cockers love this formula of chicken an...	My two cocker love formula chicken sweet potat...
Ive hunted high and low and have tried every t...	ive hunt high low tri everi type licoric toffe...
We have a rapidly growing 7 month Labradoodle ...	We rapidli grow month labradoodl puppi initi b...
we received this coffee yesterday, and have to...	receiv coffe yesterday tell love espresso n't ...

Using NLTK, the text preprocessing function performs several steps: converting to lowercase, tokenizing words, removing punctuation and stop words, excluding numbers, and applying stemming. This prepares the text for analysis, simplifying it for consistent feature extraction in machine learning models.

Score Encoding

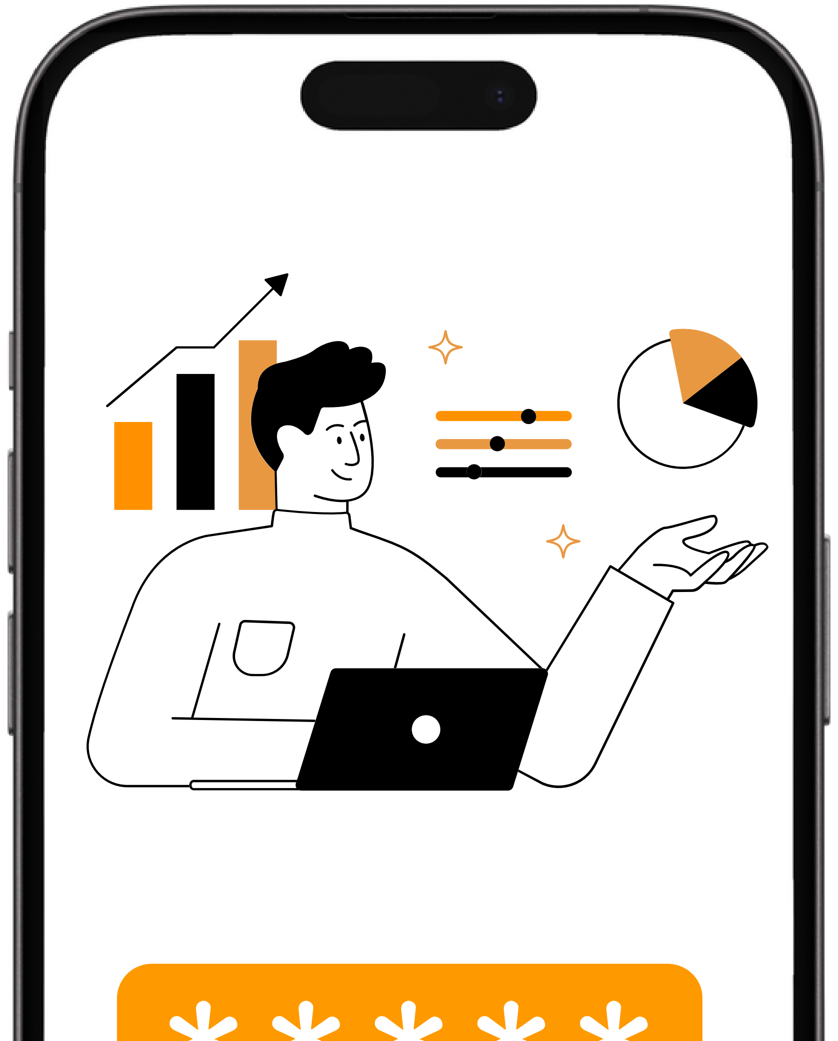
```
new_df['Score'] = new_df['Score'].apply(lambda x: 1 if x >=3 else 0)
# 1 --> Good
#0 --> Bad
```

To improve model performance, we simplified the score classification. Initially, the 5-class classification (very bad, bad, good, very good, excellent) achieved an accuracy of only 47.9% using Logistic Regression. To address this, we applied binary classification: scores of 4 and 5 were labeled as Positive (1), and scores of 1, 2, and 3 as Negative (0). This binary approach better suited the sentiment analysis task.



MODELING APPROACHES

1. Traditional models (Logistic Regression,Naive Bayes,SVM)
2. Deep Learning (LSTM)
3. Hugging Face



Logistic Regression

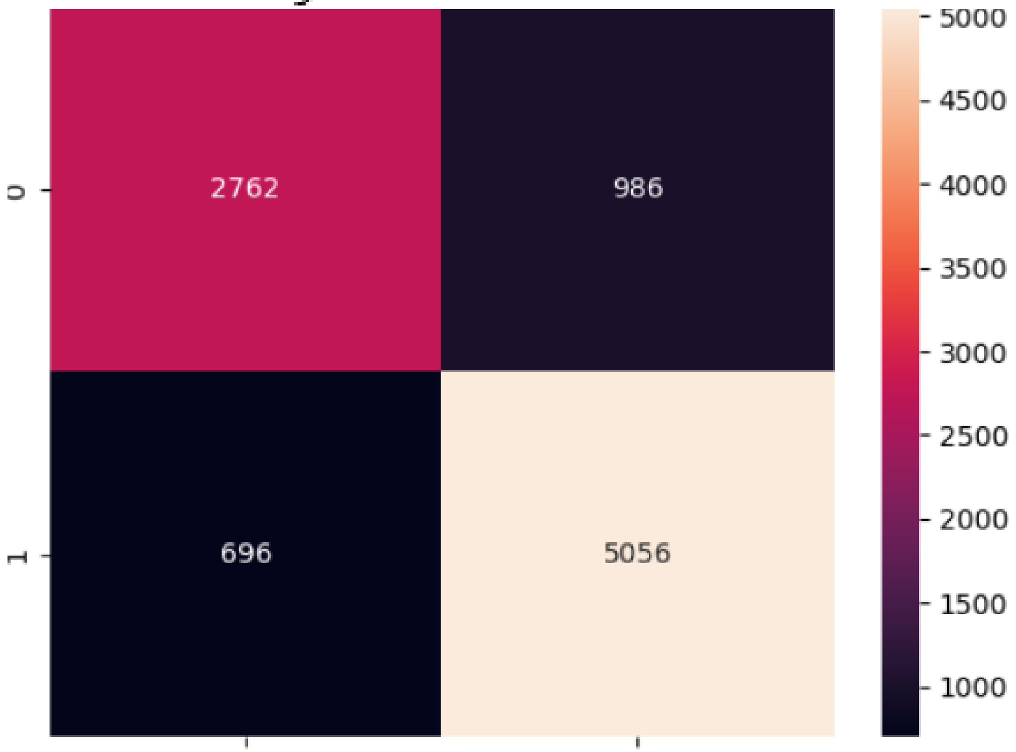
```
pipe = Pipeline(
    [
        ('vec', CountVectorizer(stop_words= "english")),
        ('tfidf', TfidfTransformer()),
        ('classifier', LogisticRegression()),
    ])
```

Training accuracy: 0.8317

Test accuracy: 0.7953

Training accuracy: 0.8591315789473685

Test accuracy: 0.8229473684210526



Logistic Regression was implemented in two configurations: the left side shows results using only review text, while the right side includes both text and summary. The pipeline outlines the key steps in model training. Combining text with the summary led to a noticeable improvement in performance, boosting accuracy by 3%, from 79.53% to 82.29%.

Naive **Bayes**

```
naive_bayes_pipeline = Pipeline([
    ('vec', CountVectorizer(stop_words='english')),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])
```

	precision	recall	f1-score	support
0	0.8427	0.3717	0.5158	4036
1	0.6915	0.9531	0.8015	5964
accuracy			0.7184	10000
macro avg	0.7671	0.6624	0.6586	10000
weighted avg	0.7525	0.7184	0.6862	10000

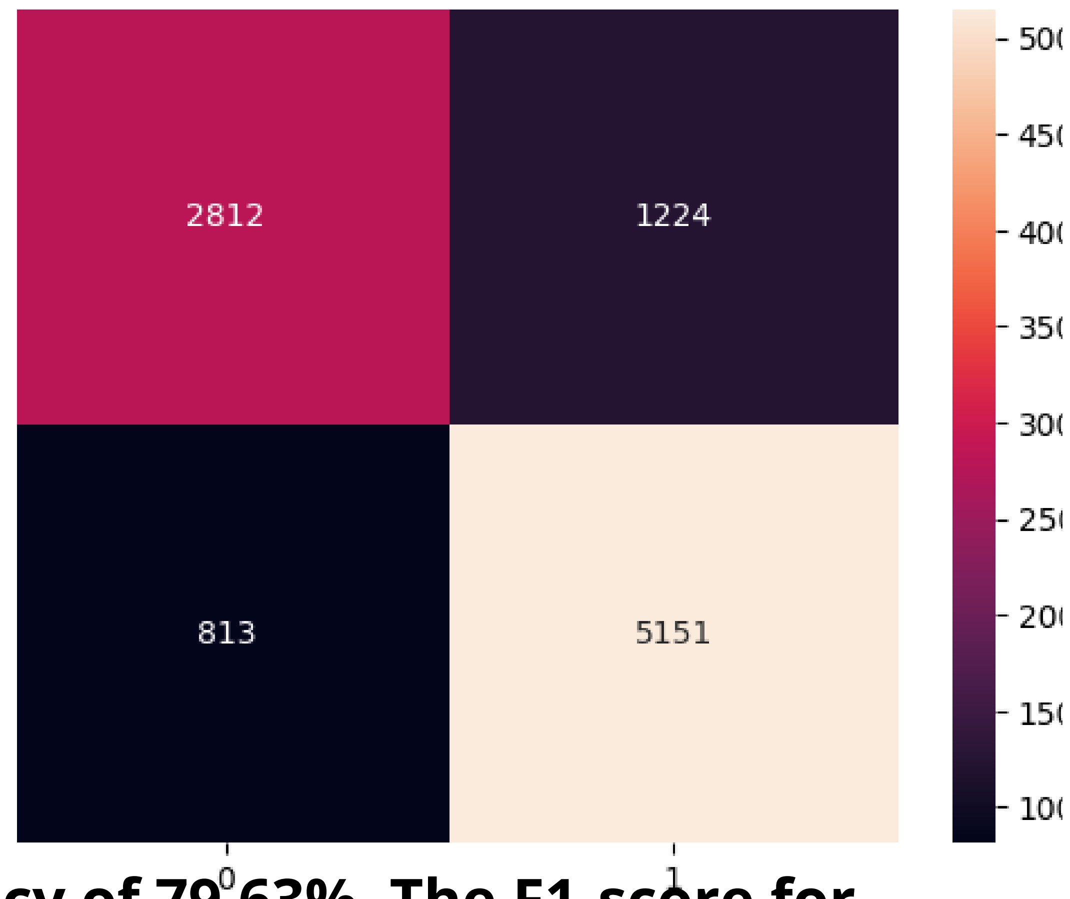
	precision	recall	f1-score	support
0	0.7987	0.7369	0.7666	3748
1	0.8368	0.8790	0.8574	5752
accuracy			0.8229	9500
macro avg	0.8178	0.8080	0.8120	9500
weighted avg	0.8218	0.8229	0.8216	9500

As shown, there is a significant difference between the two results. When using only the review text, Naive Bayes achieved 71.84% accuracy. However, by combining the text with the summary, the accuracy improved dramatically to 82.29%. This highlights the benefit of incorporating additional information for better model performance

SVC

```python
svm_pipeline = Pipeline([
    ('vec', CountVectorizer(stop_words='english')),
    ('tfidf', TfidfTransformer()),
    ('classifier', SVC())
])
```

	precision	recall	f1-score	support
0	0.7757	0.6967	0.7341	4036
1	0.8080	0.8637	0.8349	5964
accuracy			0.7963	10000
macro avg	0.7919	0.7802	0.7845	10000
weighted avg	0.7950	0.7963	0.7942	10000



The SVM model was applied using only the review text, achieving an overall accuracy of 79.63%. The F1-score for the positive class (1) was 0.8349, while the negative class (0) had a lower F1-score of 0.7341. These results show that SVM performed well in identifying positive sentiments but was less effective for negative ones.

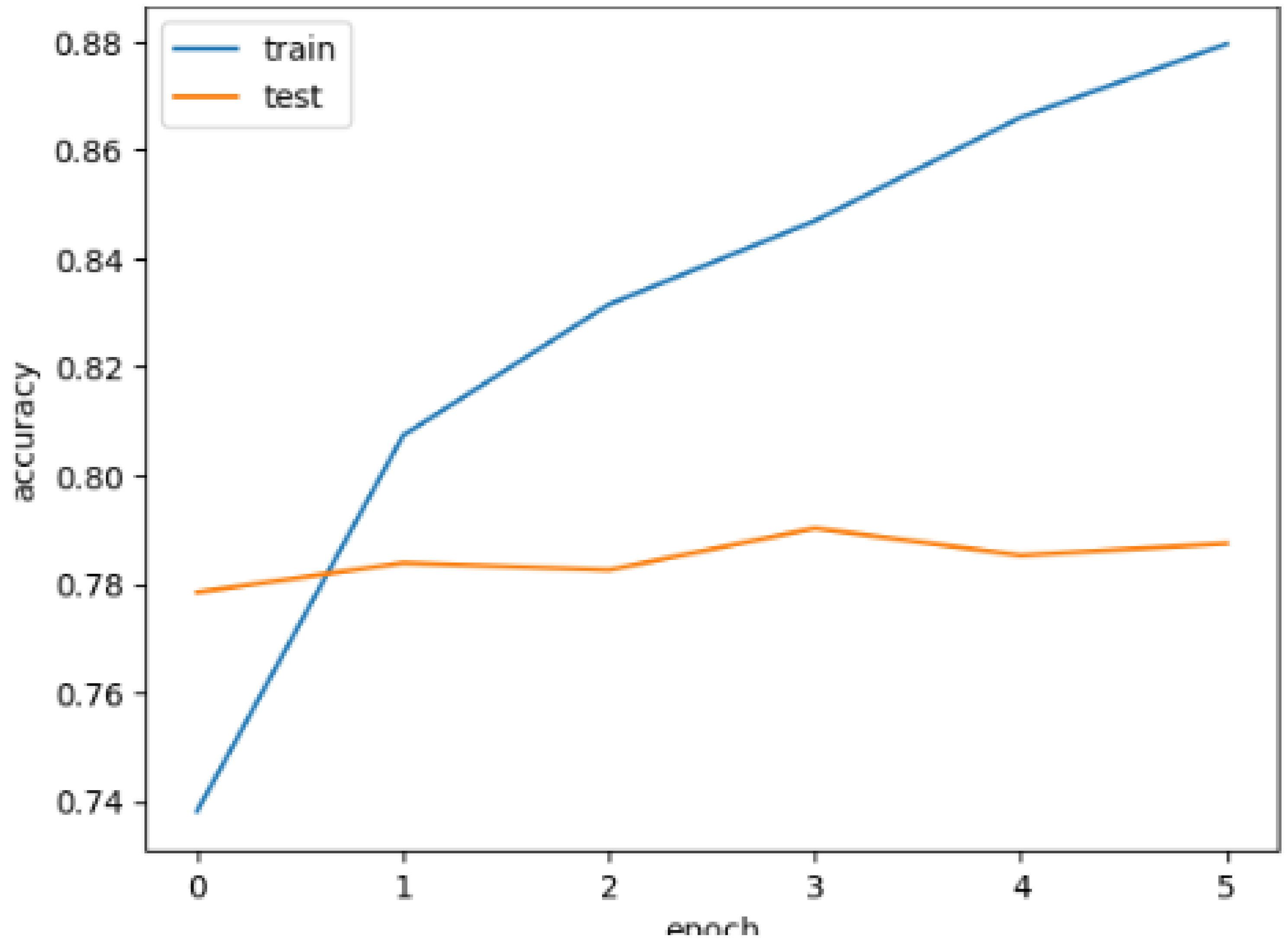
LSTM Model Architecture

The LSTM model consists of an embedding layer with a vocabulary size of 20,000 and output embedding dimension of 200, followed by an LSTM layer with 256 units, recurrent dropout (0.3), and standard dropout (0.3). A final Dense layer with a sigmoid activation function is used for binary classification, with L2 regularization to prevent overfitting. The model is designed to process padded sequences of review text for sentiment analysis

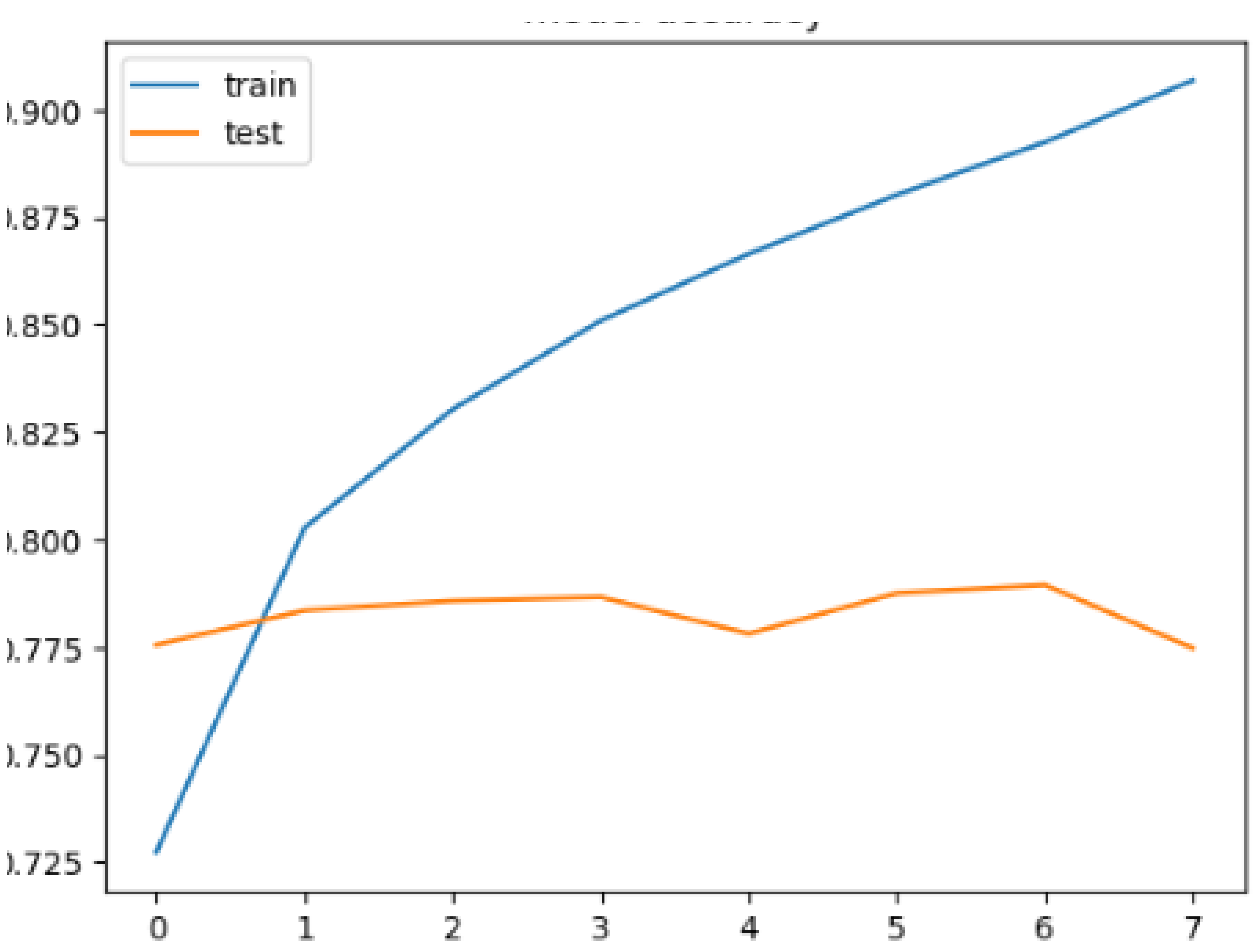
```
model=Sequential()
model.add(Embedding(input_dim=voca_size,output_dim=embedding_size,input_length=max_len))
model.add(LSTM(256,recurrent_dropout=0.3,dropout=0.3))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid',kernel_regularizer=regularizers.l2(0.01)))
```

```
# Compile the model
import tensorflow as tf
#op=tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

model_checkpoint = ModelCheckpoint(
    'best_model1.keras',  # File path where the model will be saved
    monitor='val_loss',  # Metric to monitor
    save_best_only=True,  # Save only the model with the best validation loss
    mode='min',  # 'min' because lower loss is better
    verbose=1  # Verbosity mode
)

# Train the LSTM model
history1=model.fit(X_train, y_train,
          epochs=12,
          batch_size=128,
          validation_data=(X_test, y_test)
          , callbacks=[early_stopping,model_checkpoint])
```

LSTM **Accuracy**



Final Test Accuracy: 78.75%

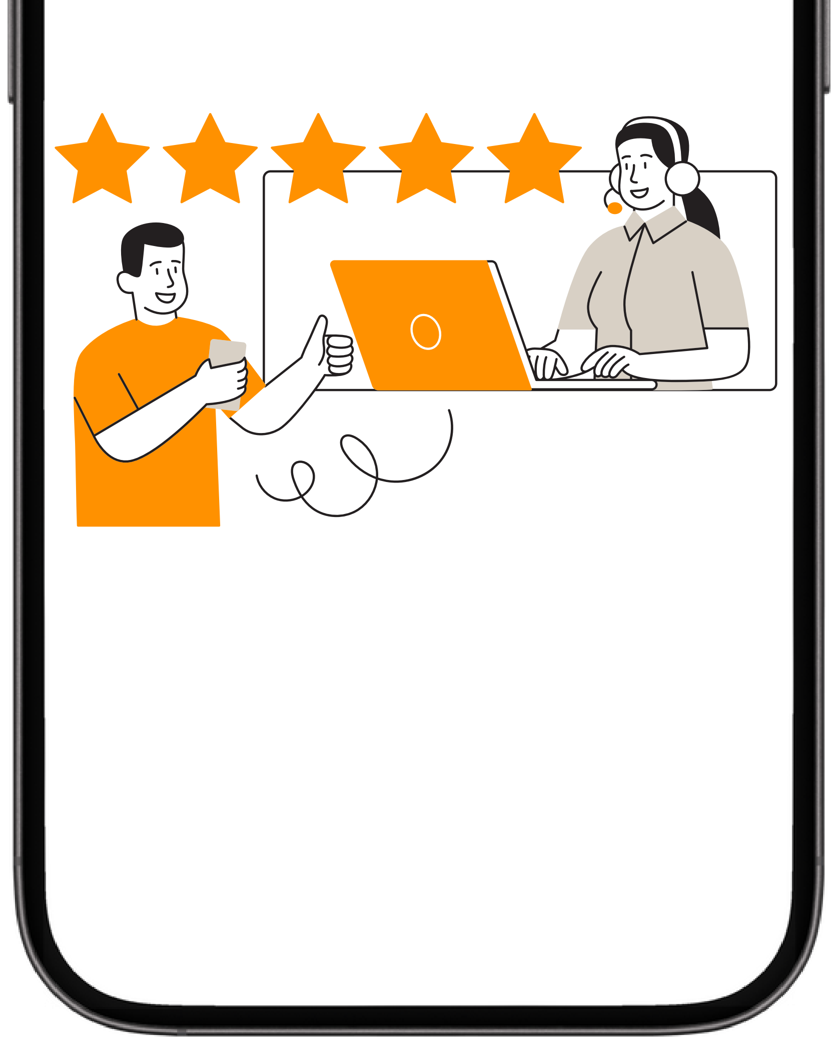


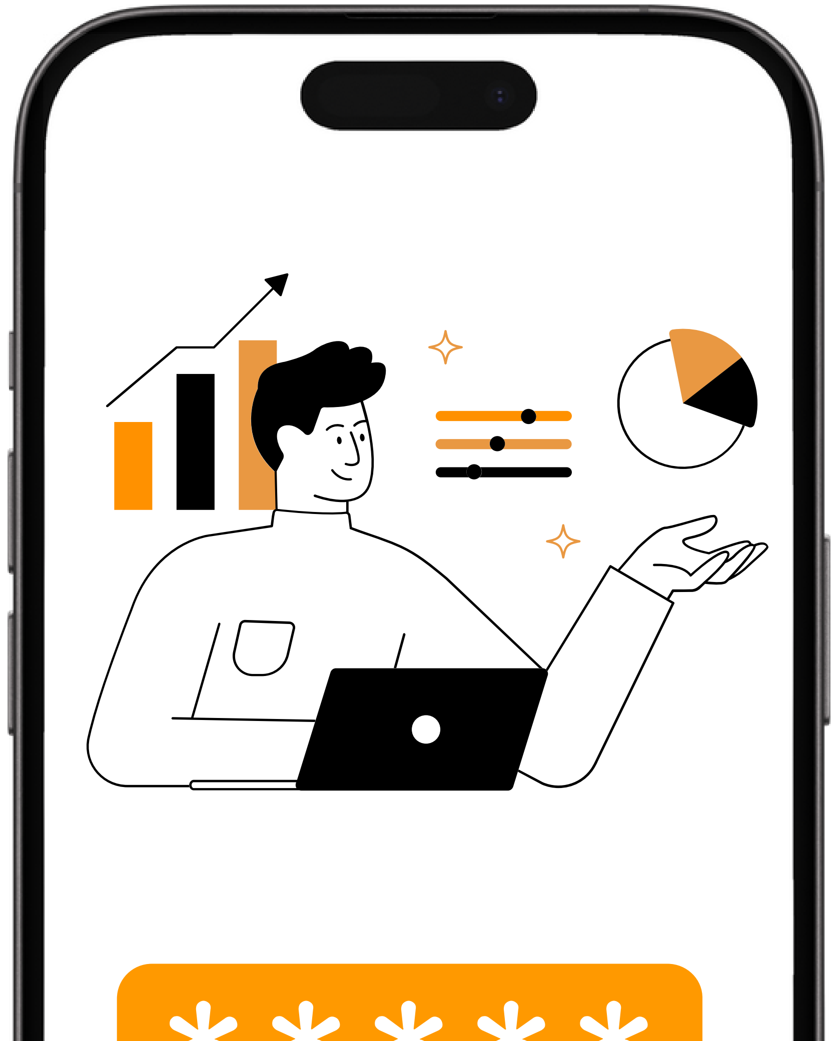
Final Test Accuracy: 77.48%

The LSTM model achieved a final test accuracy of 78.75% when trained using only the review text, showing a slight improvement in performance compared to the Text + Summary configuration.

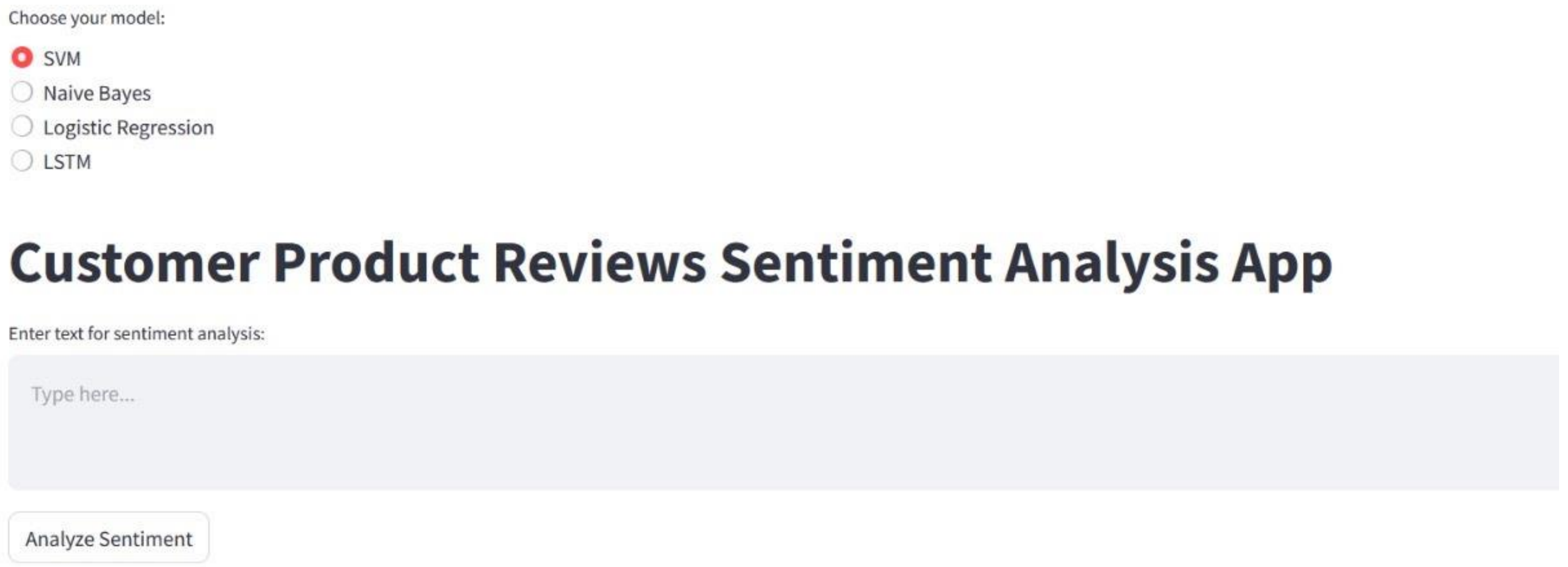
MODEL DEPLOYMENT

1.Streamlit



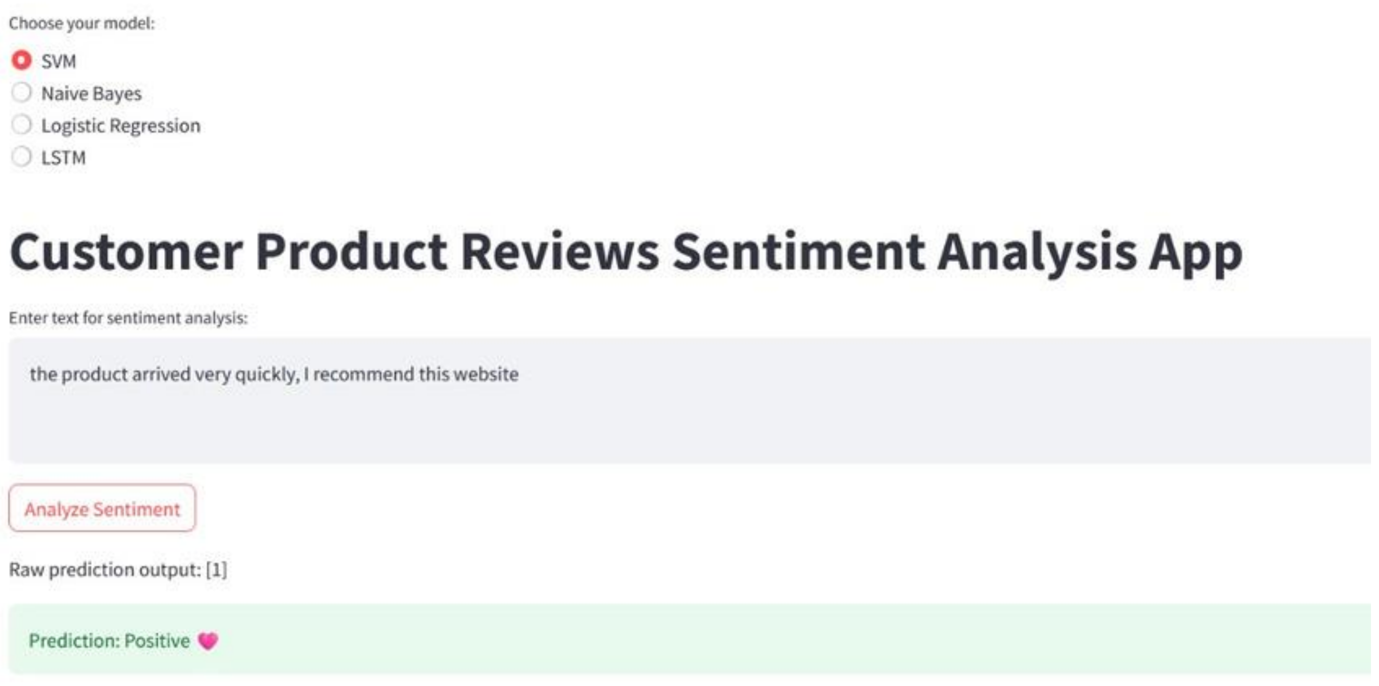


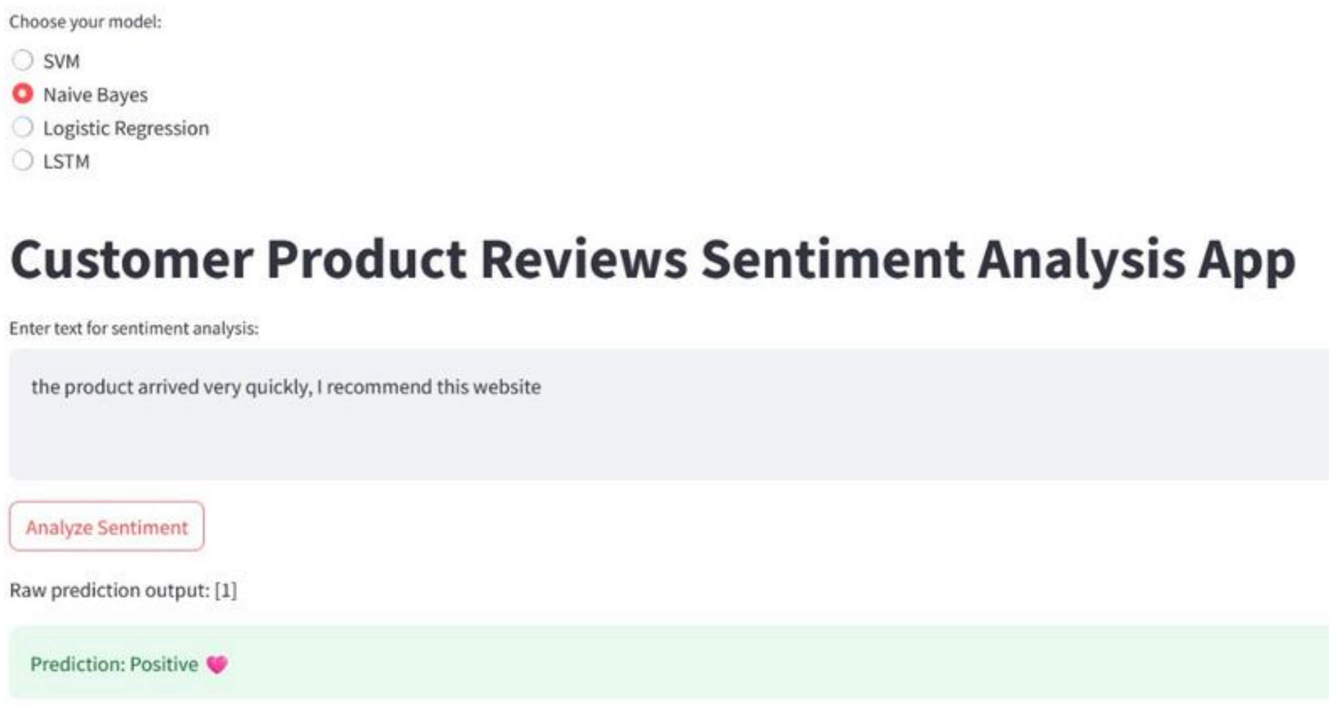
Streamlit Deployment Overview

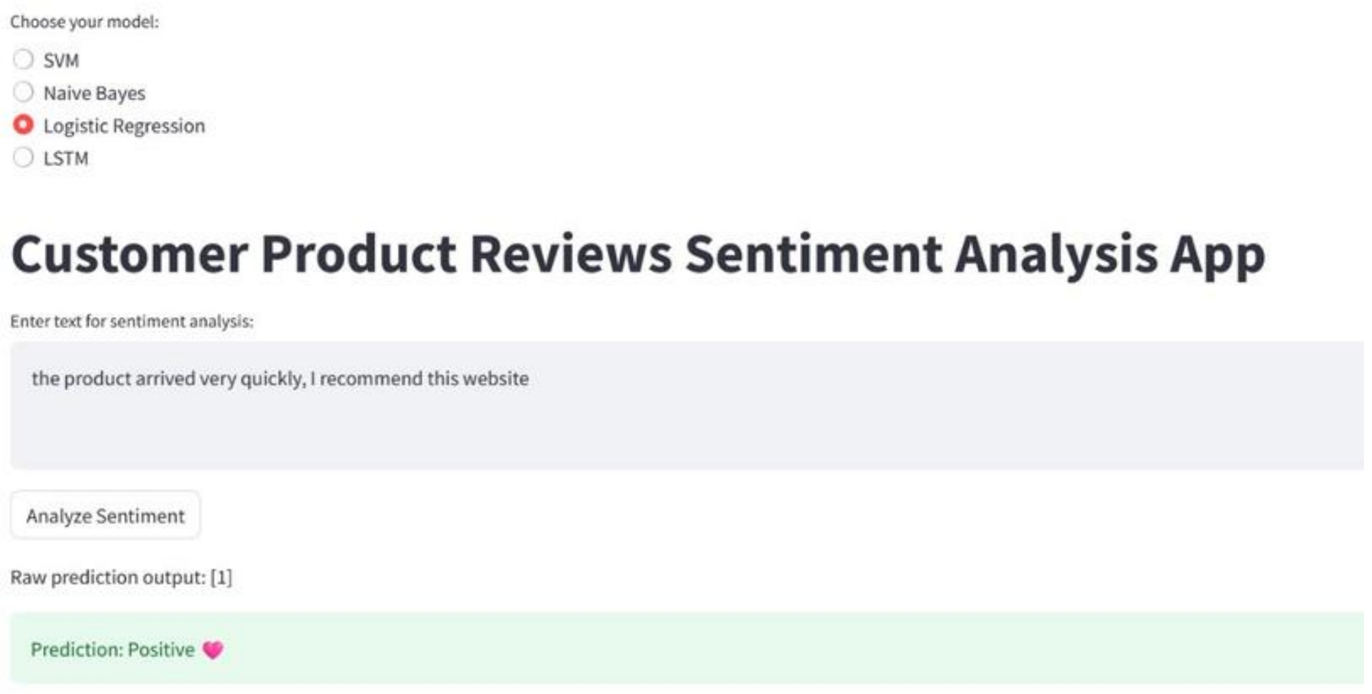


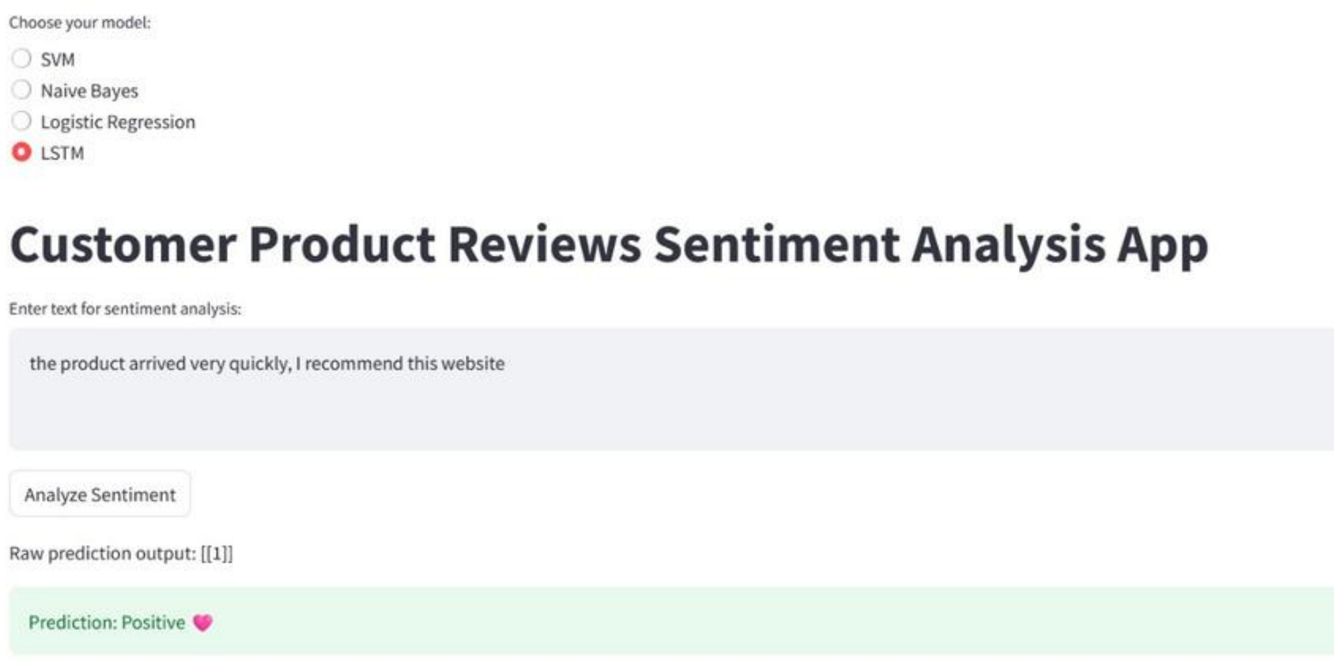
Users can input a product review and choose from four models for sentiment classification: SVM, Naive Bayes, Logistic Regression and LSTM. then predicts whether the review is Positive or Negative, displaying the raw prediction output and a visual sentiment label.

Postive Reviwes

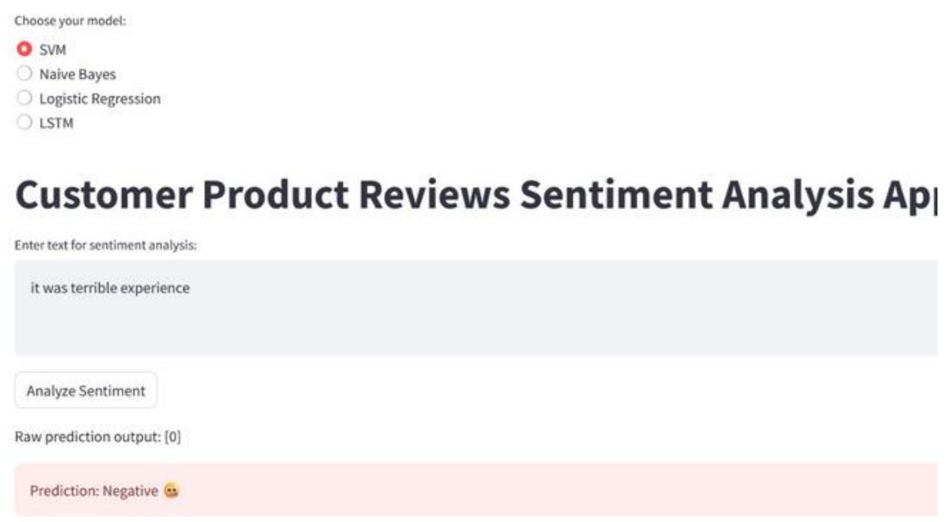


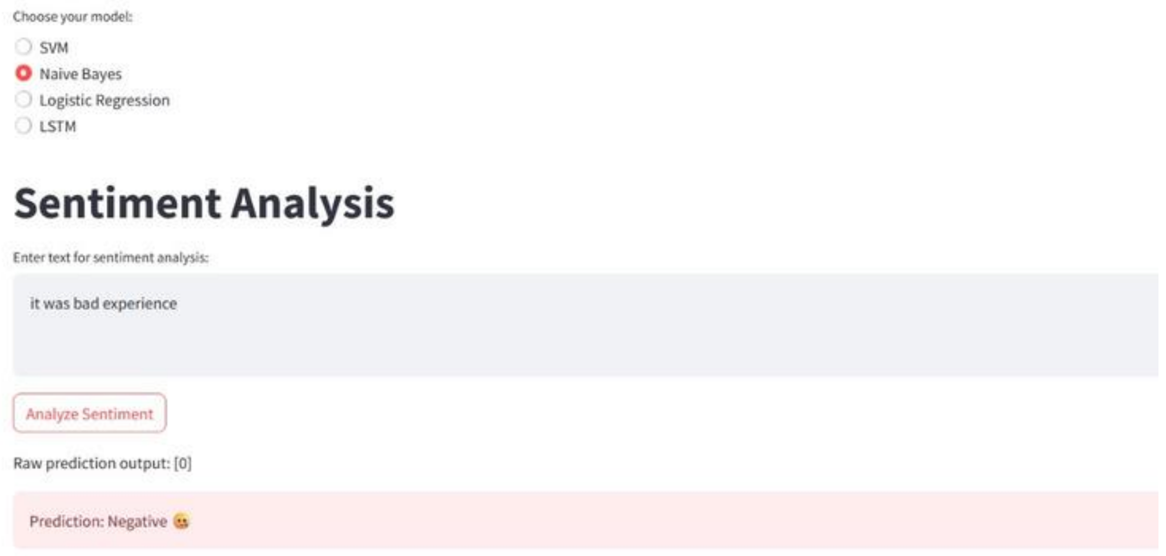


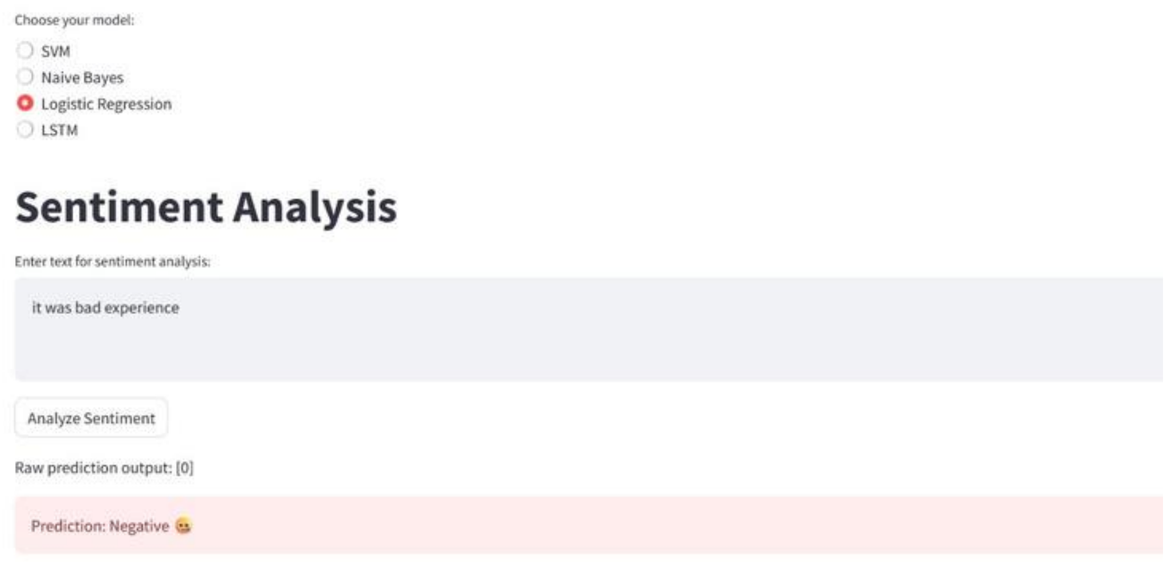


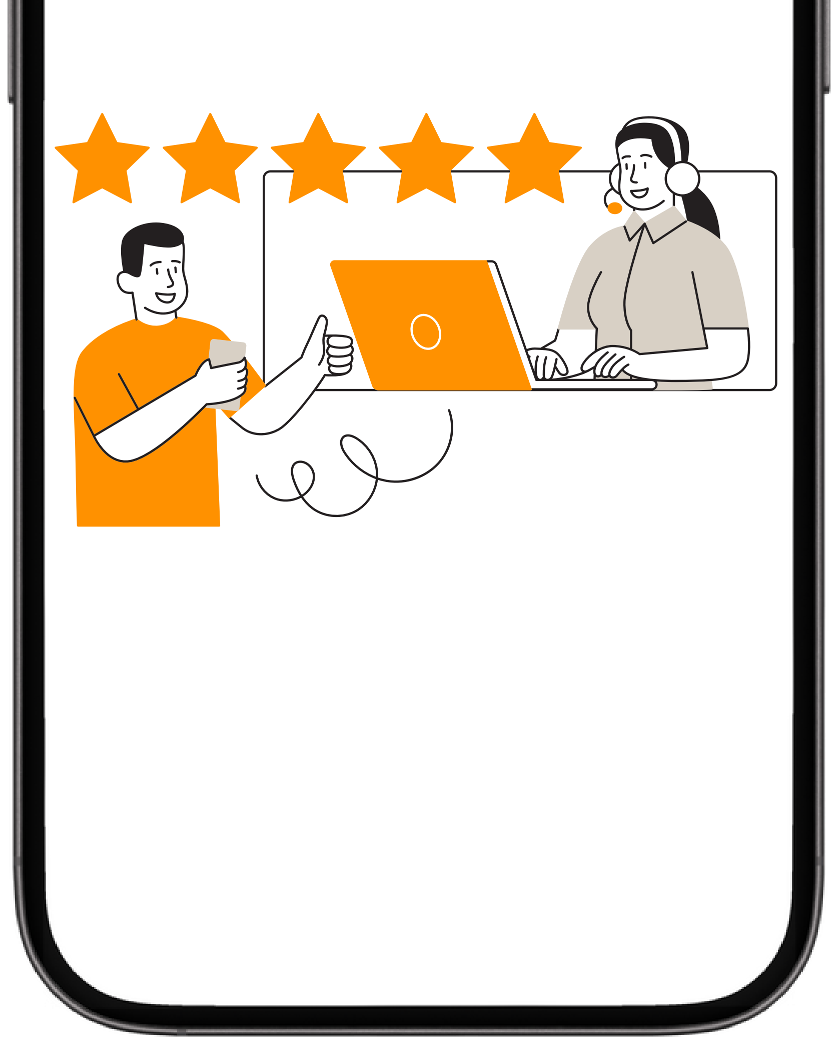


Negative Reveiws





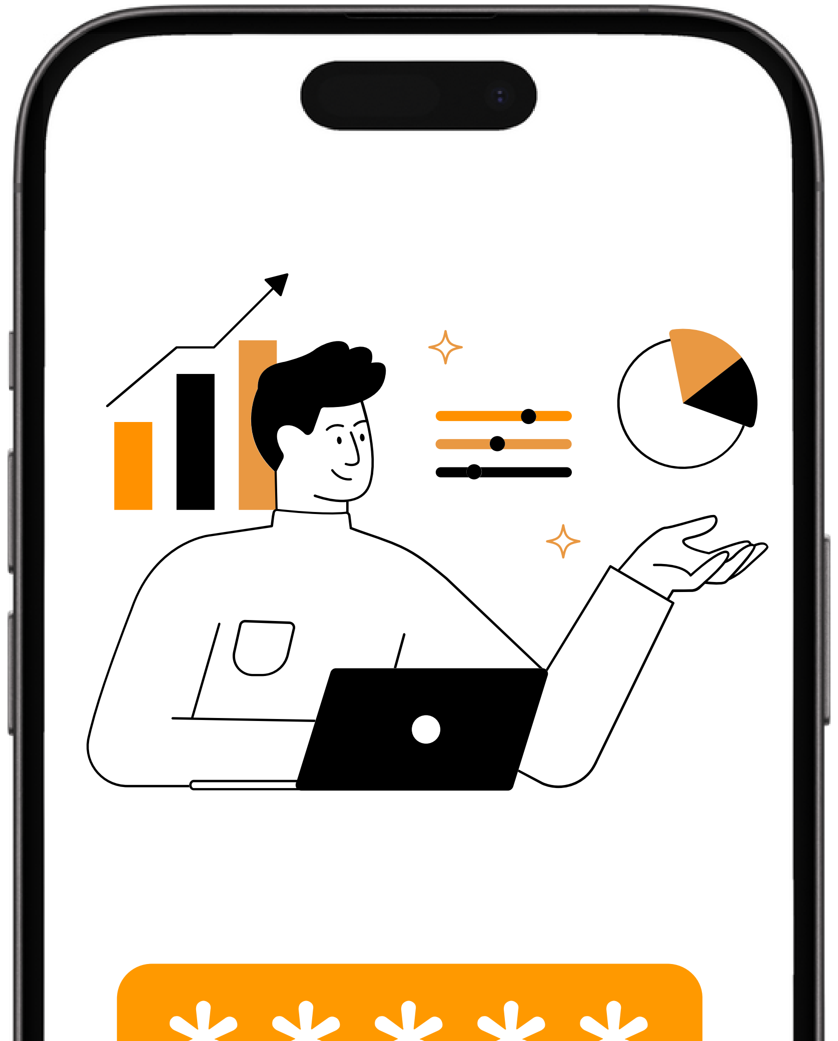




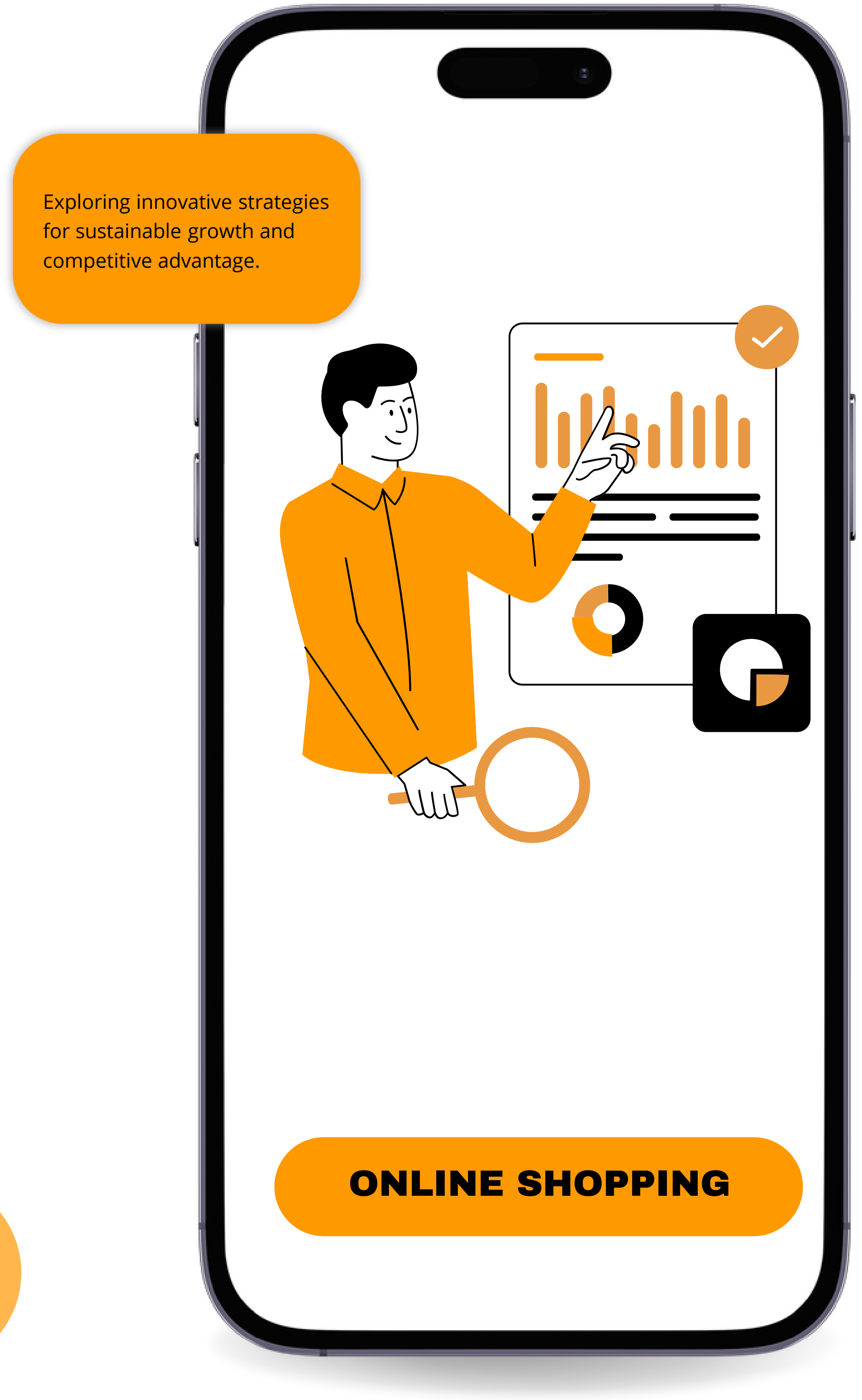
EXPERIMENT TRACKING

1.MLflow

2.Azure



Our MLflow



Machine learning tracking with MLflow

detection_with_xgb | Provide Feedback | Add Description | Share

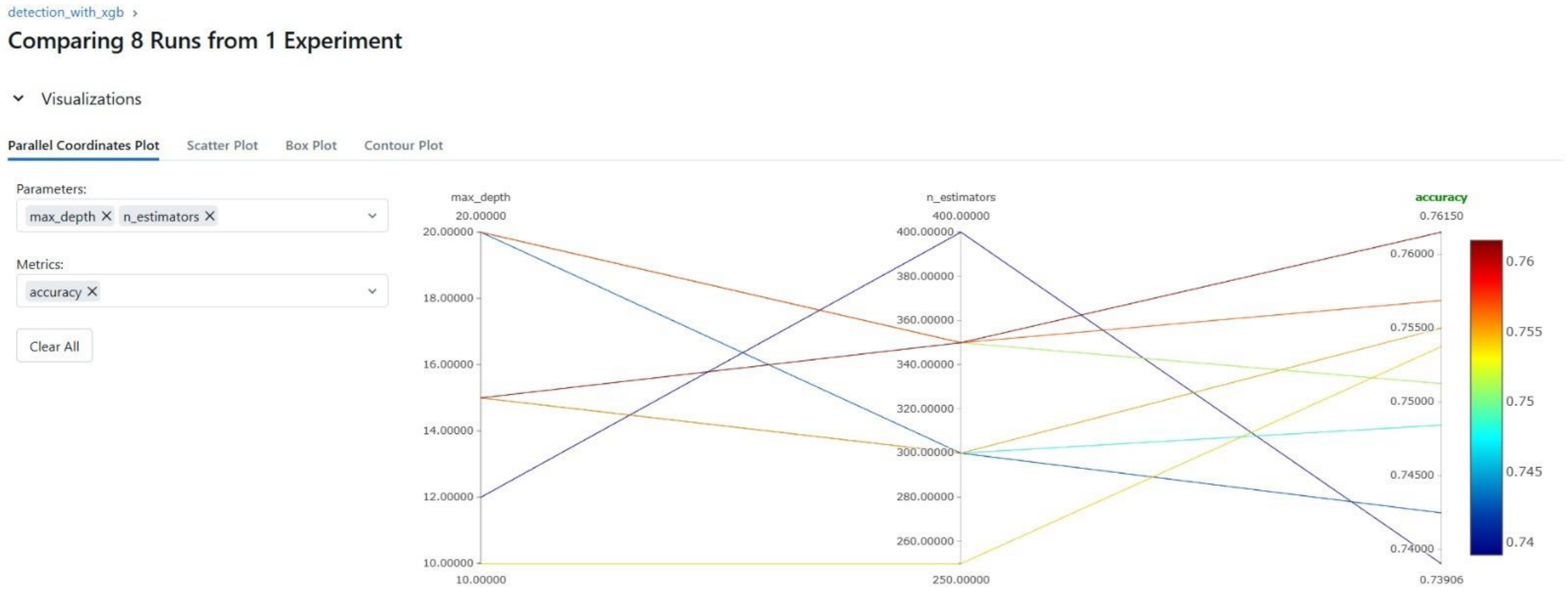
Runs | Evaluation Experimental | Traces Experimental

metrics.rmse < 1 and params.model = "tree" | Time created | State: Active | Datasets | Sort: Created | Columns | New run

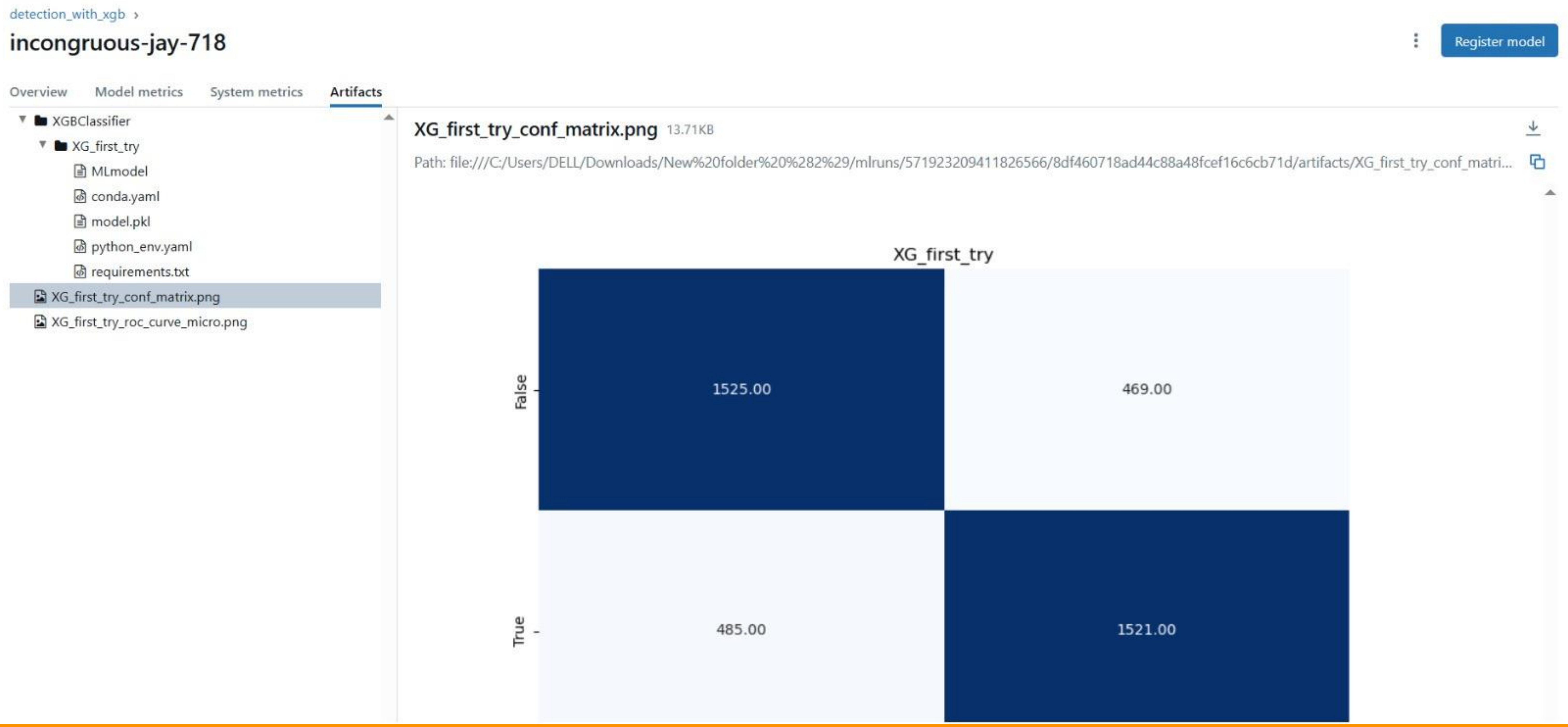
Group by

					Metrics				Parameters		Tags
Run Name	Created	Duration	Source	Models	accuracy	auc	f1-score	train_accuracy	max_depth	n_estimators	clf
powerful-fowl-112	16 minutes ago	2.2min	mlflow_...	sklearn	0.75375	0.830450135...	0.75375	0.994375	10	250	xgboost
sneaky-ray-661	26 minutes ago	4.2min	mlflow_...	sklearn	0.7425	0.822427339...	0.7425	0.999921875	20	300	xgboost
whimsical-fish-275	40 minutes ago	3.7min	mlflow_...	sklearn	0.7390625	0.822898855...	0.7390625	0.999921875	12	400	xgboost
welcoming-zebra-486	3 hours ago	3.3min	mlflow_...	sklearn	0.7484375	0.831002515...	0.7484375	0.999921875	15	300	xgboost
colorful-shrimp-339	4 hours ago	3.8min	mlflow_...	sklearn	0.755	0.842836499...	0.755	1	15	300	xgboost
zealous-sloth-506	4 hours ago	4.9min	mlflow_...	sklearn	0.75125	0.834254683...	0.75125	1	20	350	xgboost
powerful-deer-391	4 hours ago	5.1min	mlflow_...	sklearn	0.756875	0.832760248...	0.756875	0.999921875	20	350	xgboost
incongruous-jay-718	5 hours ago	4.9min	mlflow_...	sklearn	0.7615	0.843707843...	0.7615	0.999875	15	350	xgboost

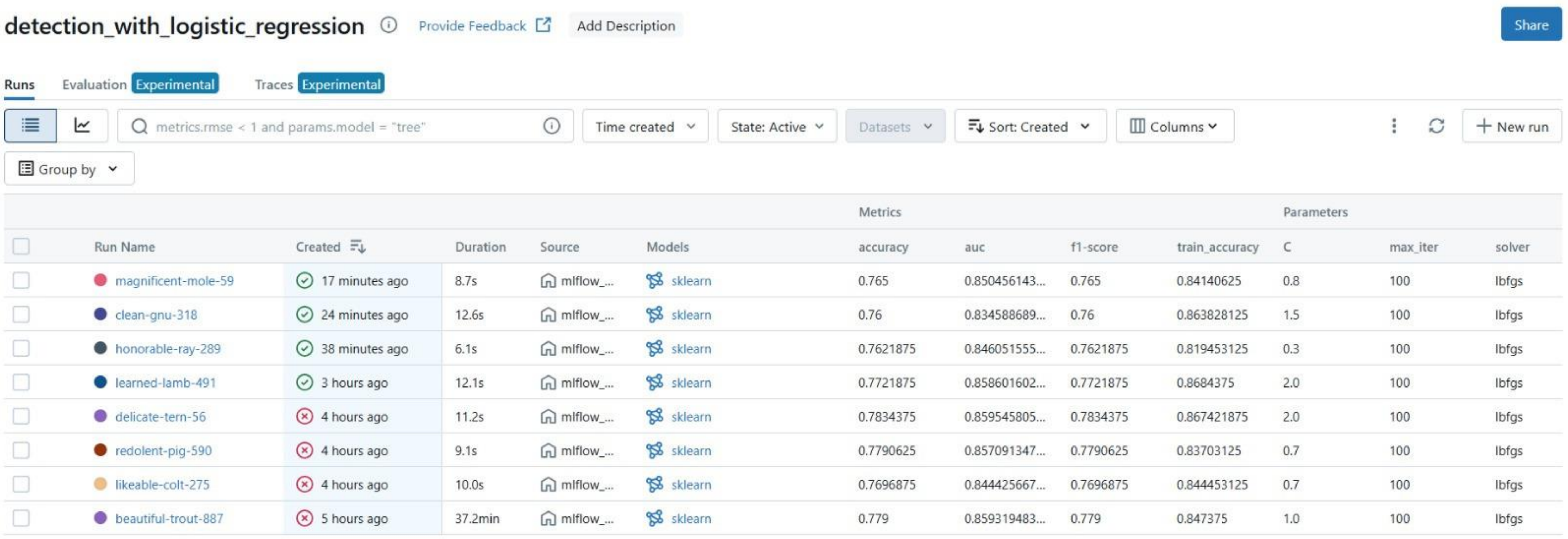
XGBOOST EXPERIMENT



COMPARING XGBOOST

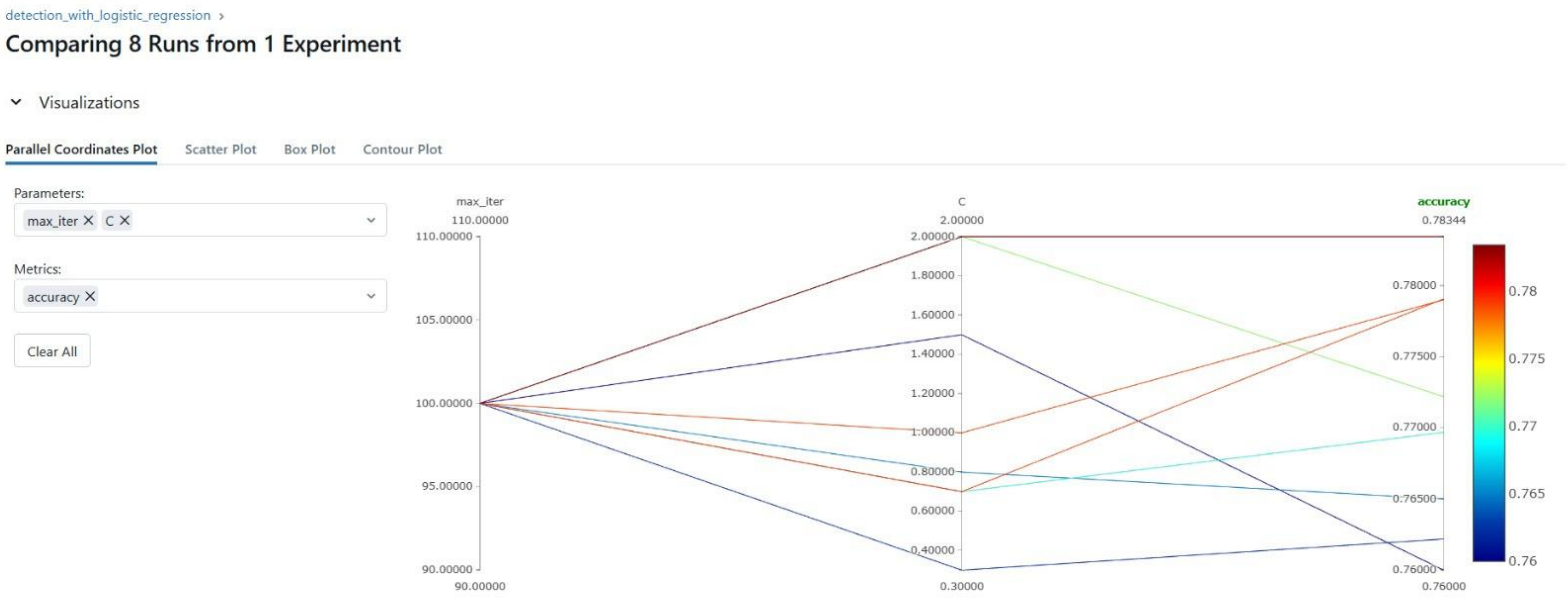


XGBOOST CONFUSION MATRIX

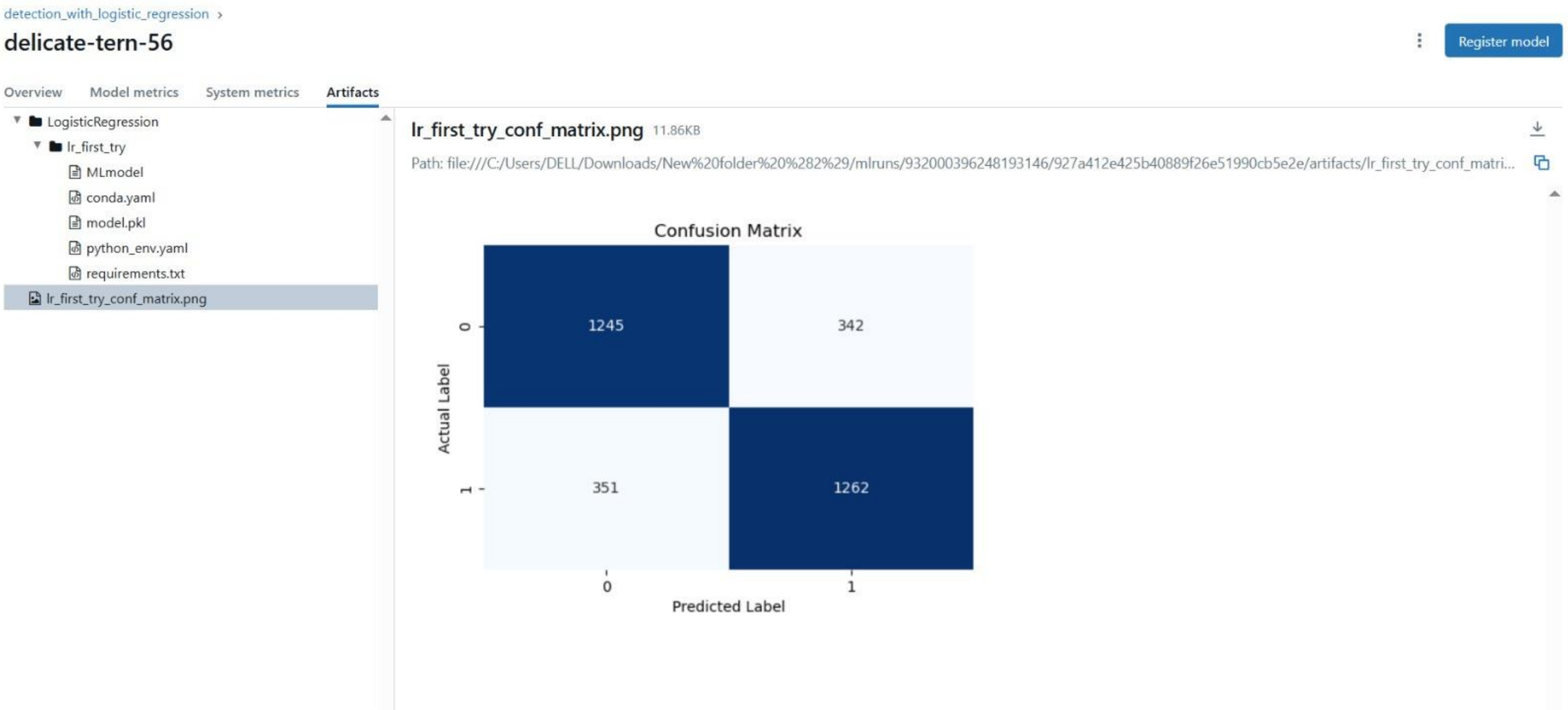


						Metrics				Parameters		
	Run Name	Created	Duration	Source	Models	accuracy	auc	f1-score	train_accuracy	C	max_iter	solver
	magnificent-mole-59	17 minutes ago	8.7s	mlflow_...	sklearn	0.765	0.850456143...	0.765	0.84140625	0.8	100	lbfgs
	clean-gnu-318	24 minutes ago	12.6s	mlflow_...	sklearn	0.76	0.834588689...	0.76	0.863828125	1.5	100	lbfgs
	honorable-ray-289	38 minutes ago	6.1s	mlflow_...	sklearn	0.7621875	0.846051555...	0.7621875	0.819453125	0.3	100	lbfgs
	learned-lamb-491	3 hours ago	12.1s	mlflow_...	sklearn	0.7721875	0.858601602...	0.7721875	0.8684375	2.0	100	lbfgs
	delicate-tern-56	4 hours ago	11.2s	mlflow_...	sklearn	0.7834375	0.859545805...	0.7834375	0.867421875	2.0	100	lbfgs
	redolent-pig-590	4 hours ago	9.1s	mlflow_...	sklearn	0.7790625	0.857091347...	0.7790625	0.83703125	0.7	100	lbfgs
	likeable-colt-275	4 hours ago	10.0s	mlflow_...	sklearn	0.7696875	0.844425667...	0.7696875	0.844453125	0.7	100	lbfgs
	beautiful-trout-887	5 hours ago	37.2min	mlflow_...	sklearn	0.779	0.859319483...	0.779	0.847375	1.0	100	lbfgs

LOGISTIC REGRESSION

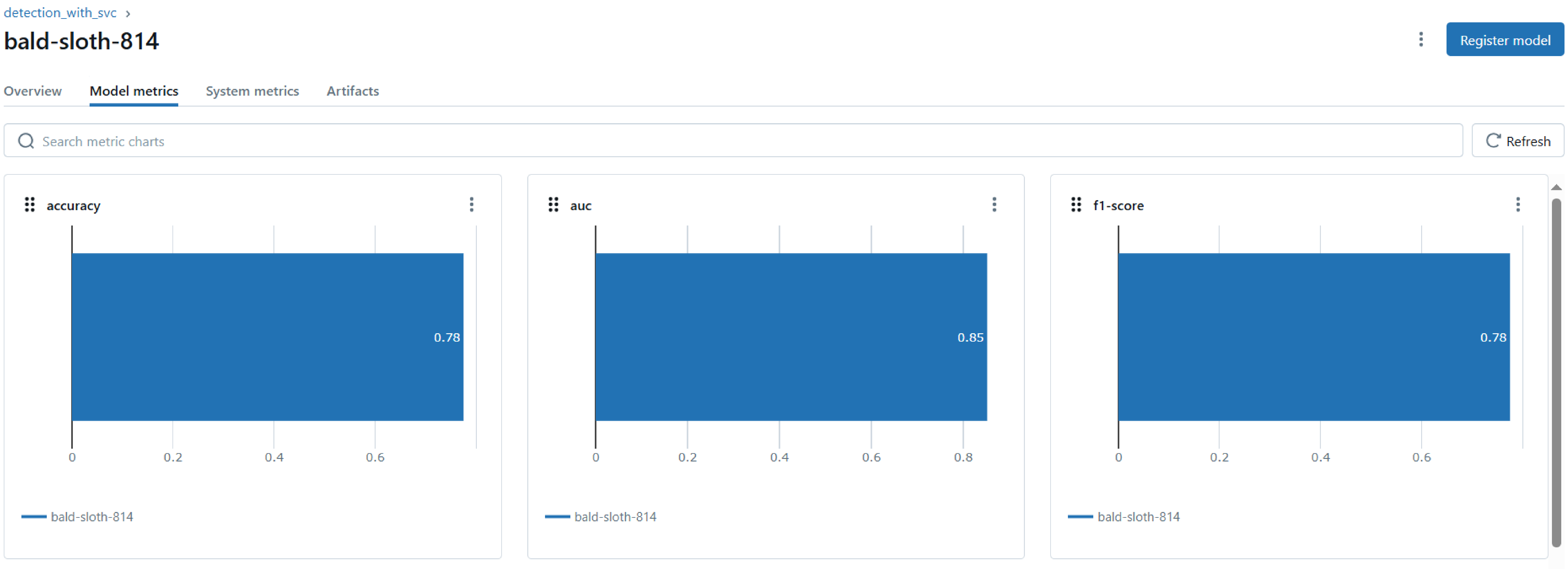


COMPARING LOGISTIC REGRESSION



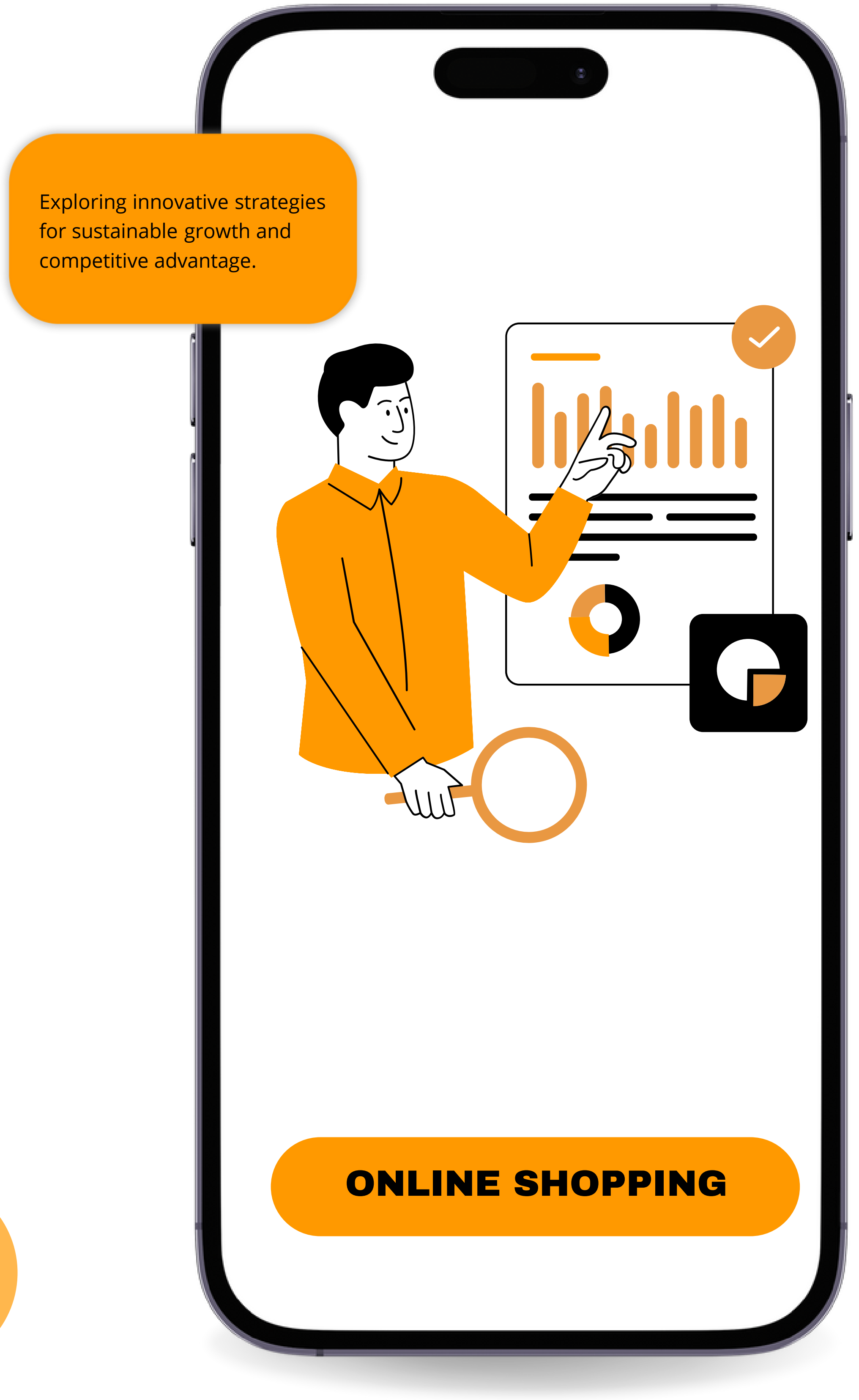
LOGISTIC REGRESSION

SVC CONFUSION MATRIX

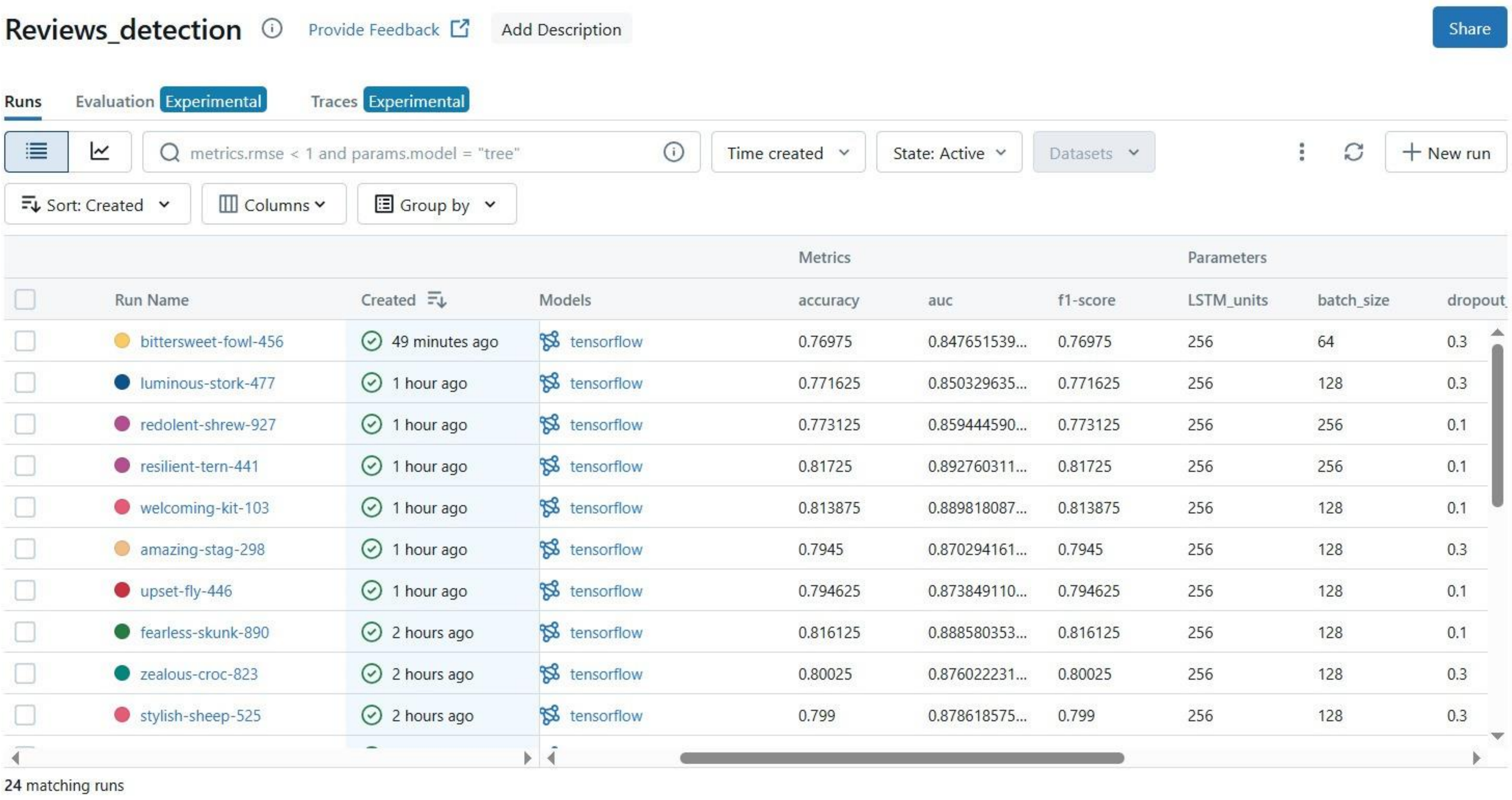

detection_with_svc ›
bald-sloth-814
Register model
Overview
Model metrics
System metrics
Artifacts
Search metric charts
Refresh
accuracy
0.78
0
0.2
0.4
0.6
bald-sloth-814
auc
0.85
0
0.2
0.4
0.6
0.8
bald-sloth-814
f1-score
0.78
0
0.2
0.4
0.6
bald-sloth-814

SVC MATRIXS

Our MLflow

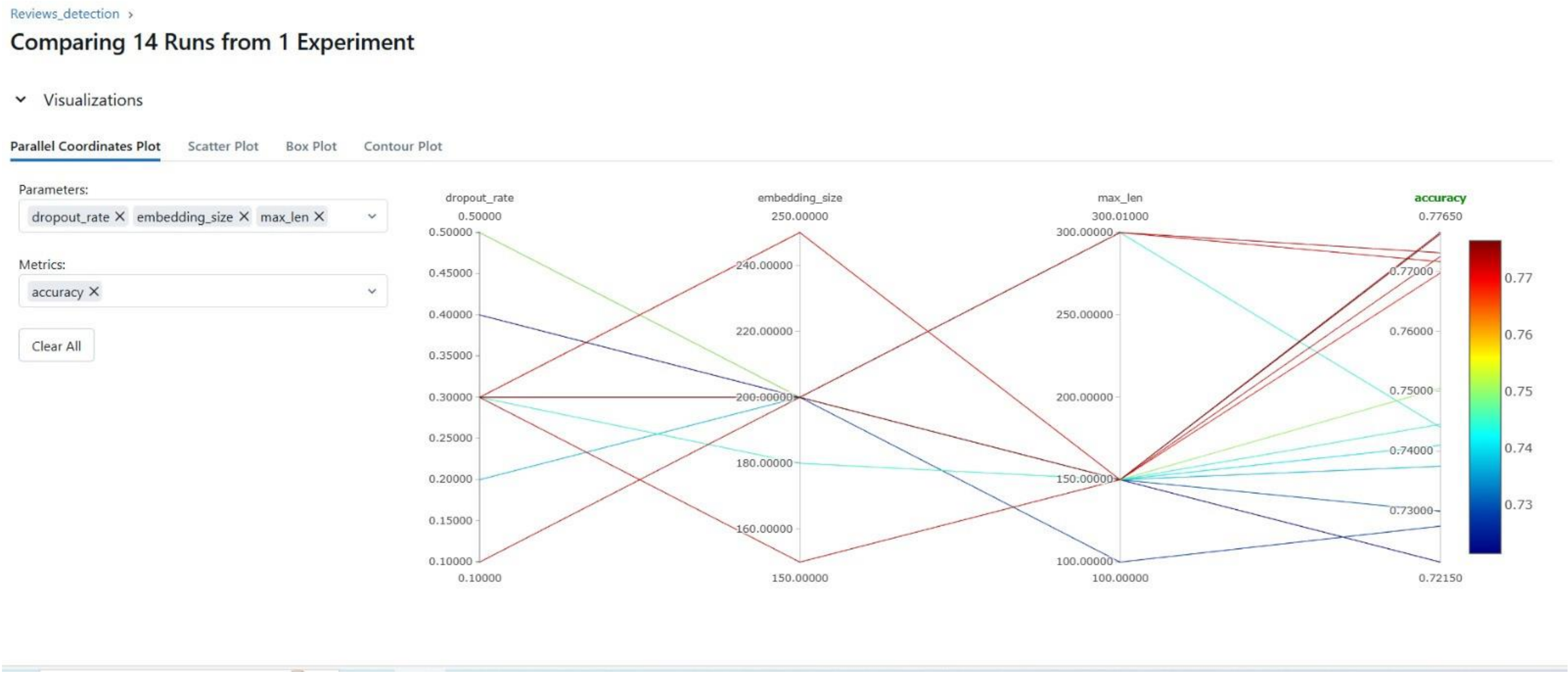


Deep learning tracking with MLflow

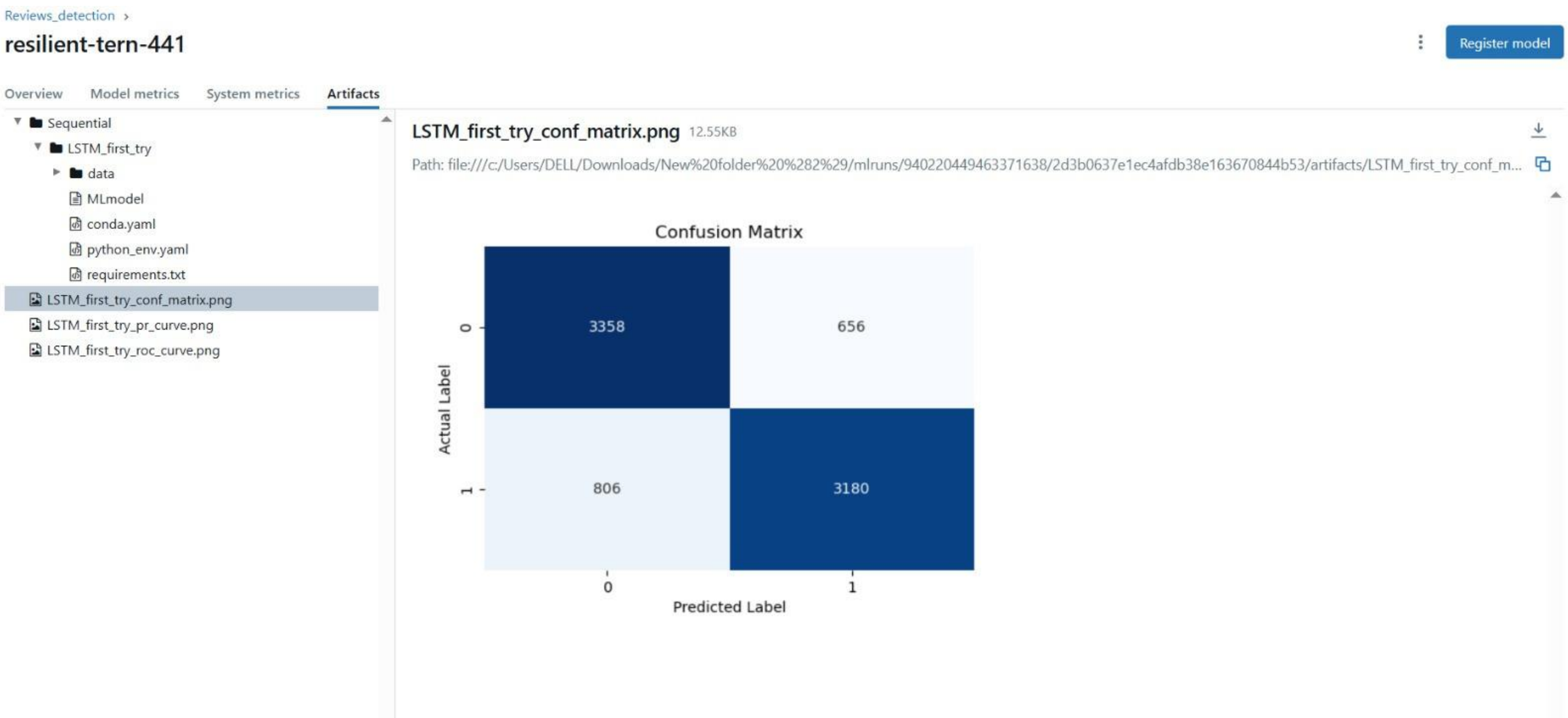


			Metrics			Parameters		
Run Name	Created	Models	accuracy	auc	f1-score	LSTM_units	batch_size	dropout_
bittersweet-fowl-456	49 minutes ago	tensorflow	0.76975	0.847651539...	0.76975	256	64	0.3
luminous-stork-477	1 hour ago	tensorflow	0.771625	0.850329635...	0.771625	256	128	0.3
redolent-shrew-927	1 hour ago	tensorflow	0.773125	0.859444590...	0.773125	256	256	0.1
resilient-tern-441	1 hour ago	tensorflow	0.81725	0.892760311...	0.81725	256	256	0.1
welcoming-kit-103	1 hour ago	tensorflow	0.813875	0.889818087...	0.813875	256	128	0.1
amazing-stag-298	1 hour ago	tensorflow	0.7945	0.870294161...	0.7945	256	128	0.3
upset-fly-446	1 hour ago	tensorflow	0.794625	0.873849110...	0.794625	256	128	0.1
fearless-skunk-890	2 hours ago	tensorflow	0.816125	0.888580353...	0.816125	256	128	0.1
zealous-croc-823	2 hours ago	tensorflow	0.80025	0.876022231...	0.80025	256	128	0.3
stylish-sheep-525	2 hours ago	tensorflow	0.799	0.878618575...	0.799	256	128	0.3

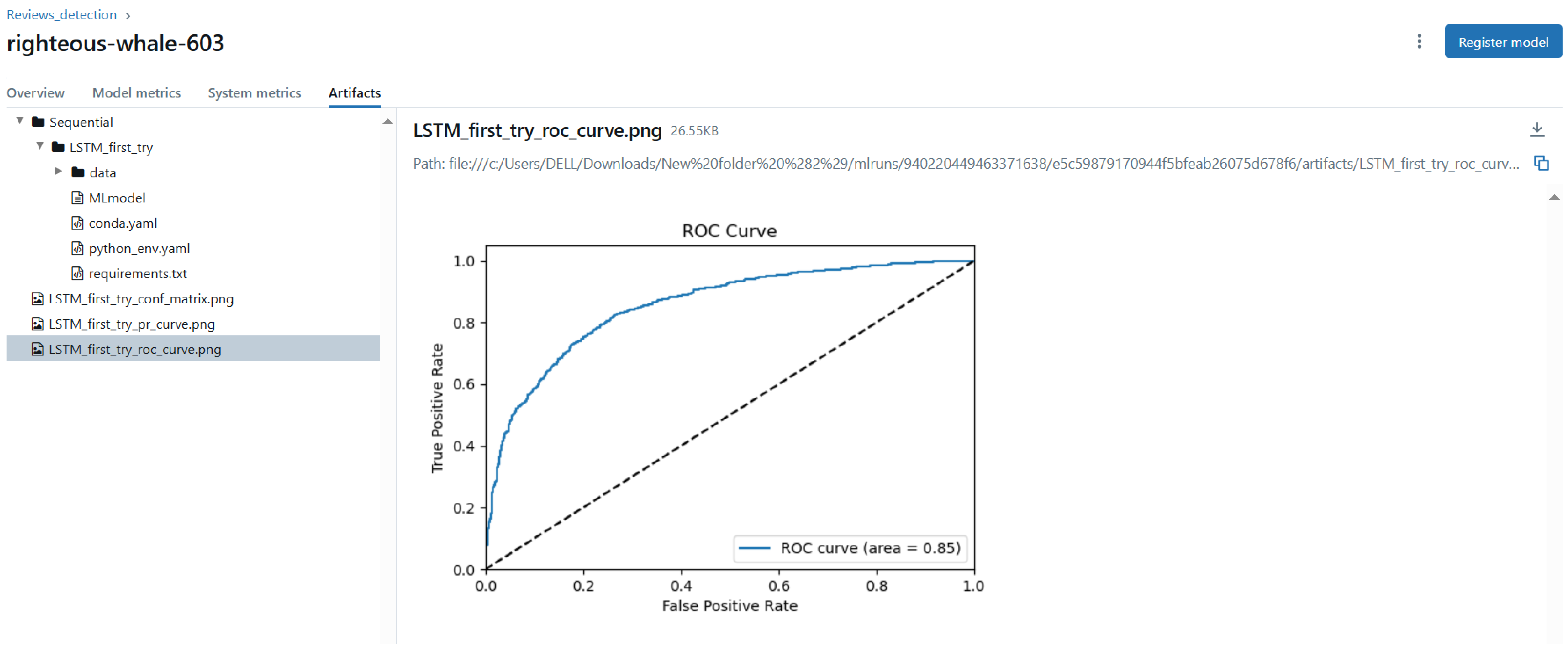
LSTM EXPERIMENT



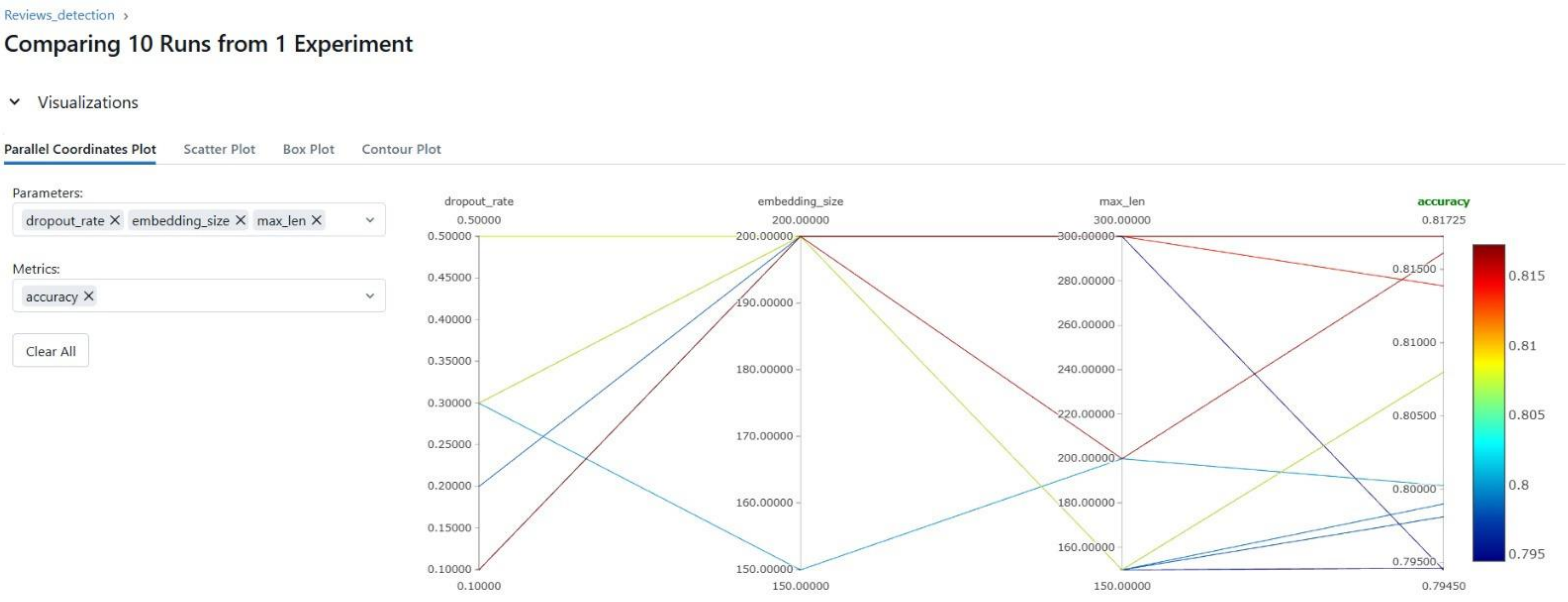
COMPARING LSTM WITH TEXT


Reviews_detection ›
resilient-tern-441
Register model
Overview
Model metrics
System metrics
Artifacts
Sequential
LSTM_first_try
data
MLmodel
conda.yaml
python_env.yaml
requirements.txt
LSTM_first_try_conf_matrix.png
LSTM_first_try_pr_curve.png
LSTM_first_try_roc_curve.png
LSTM_first_try_conf_matrix.png 12.55KB
Path: file:///c:/Users/DELL/Downloads/New%20folder%20%282%29/mlruns/940220449463371638/2d3b0637e1ec4afdb38e163670844b53/artifacts/LSTM_first_try_conf_m...
Confusion Matrix
3358
656
806
3180
Actual Label
0
1
0
1
Predicted Label

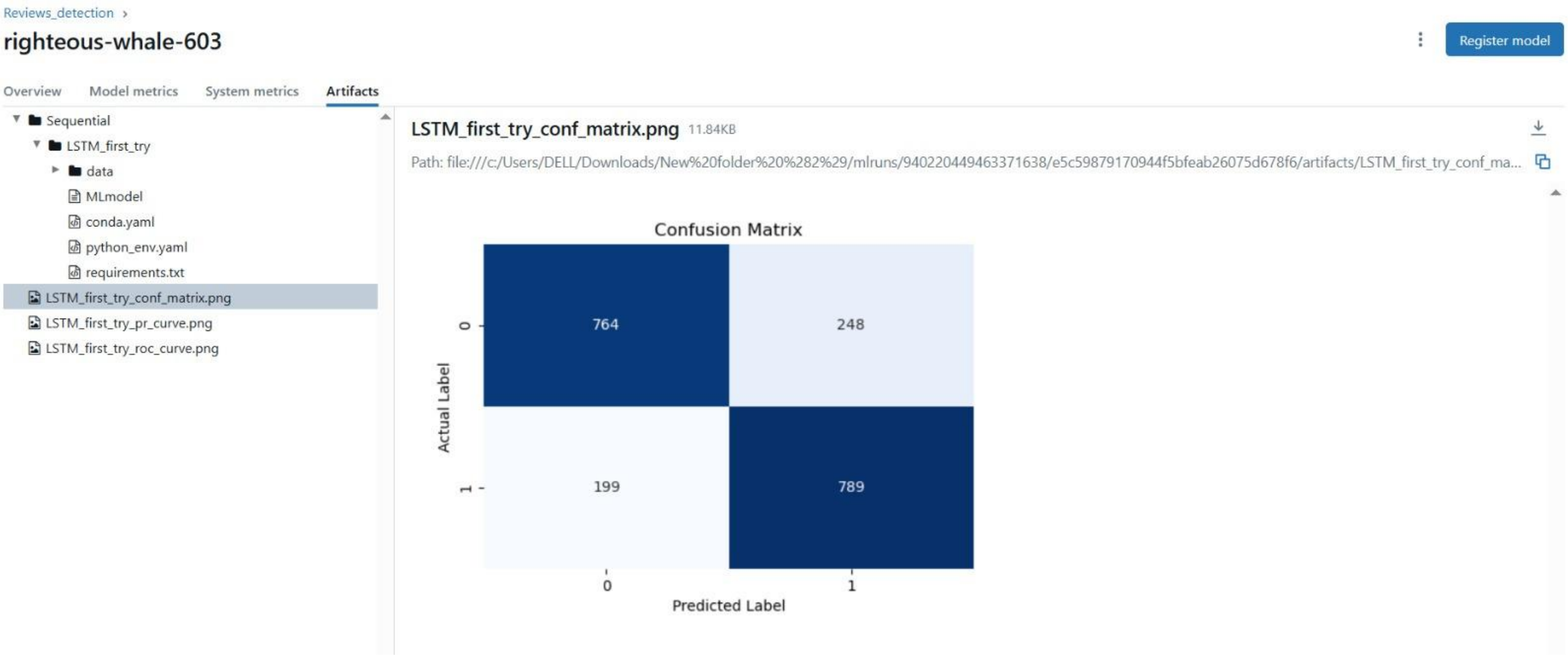
LSTM



LSTM ROC CURVE



LSTM WITH TEXT AND SUMMRY



LSTM2 CONFUION MATRIX

Reviews_detection ›

resilient-tern-441

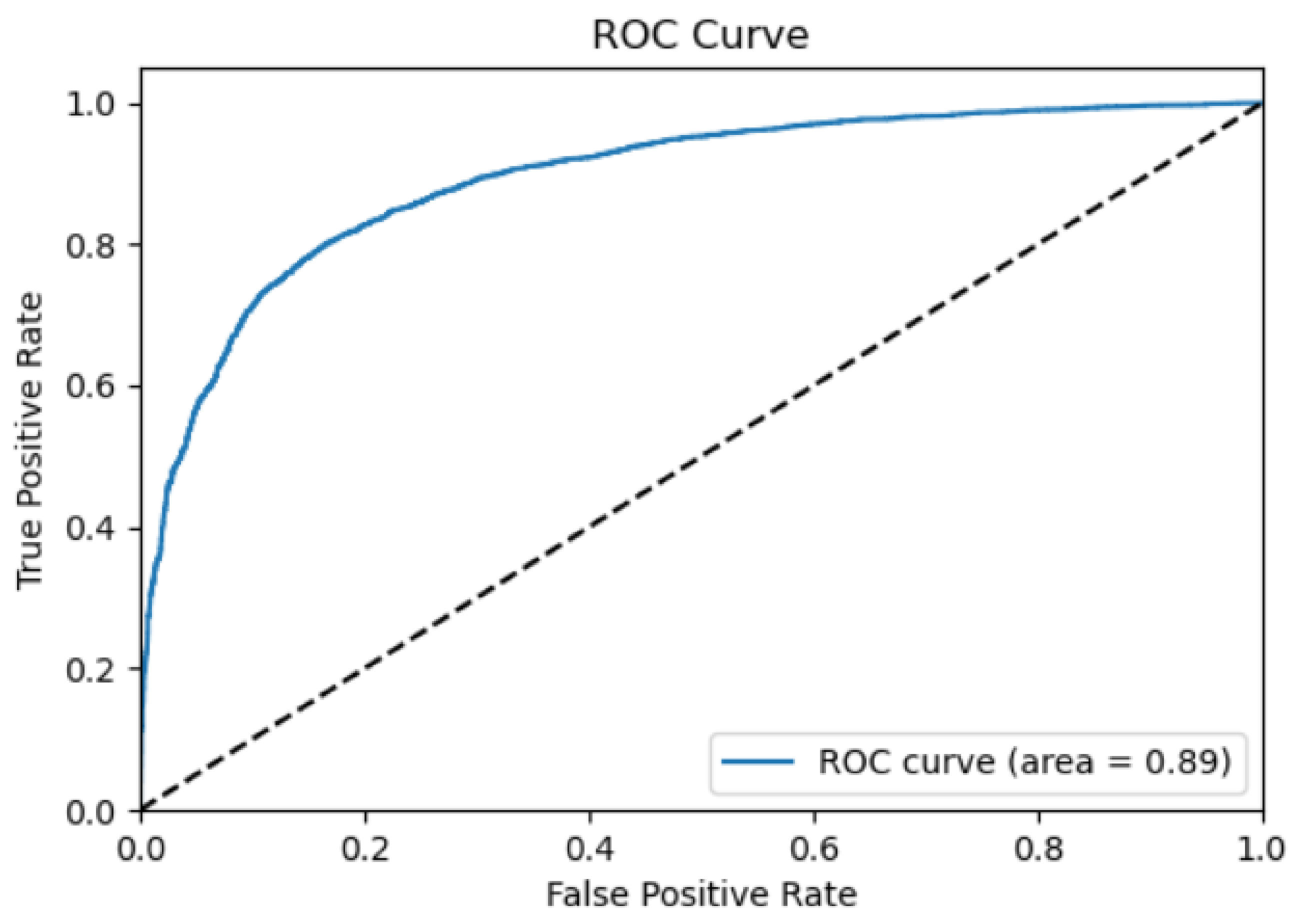
Register model

Overview Model metrics System metrics **Artifacts**

- Sequential
  - LSTM_first_try
    - data
    - MLmodel
    - conda.yaml
    - python_env.yaml
    - requirements.txt
  - LSTM_first_try_conf_matrix.png
  - LSTM_first_try_pr_curve.png
  - LSTM_first_try_roc_curve.png

LSTM_first_try_roc_curve.png 29.33KB

Path: file:///c:/Users/DELL/Downloads/New%20folder%20%282%29/mlruns/940220449463371638/2d3b0637e1ec4afdb38e163670844b53/artifacts/LSTM_first_try_roc_cur...

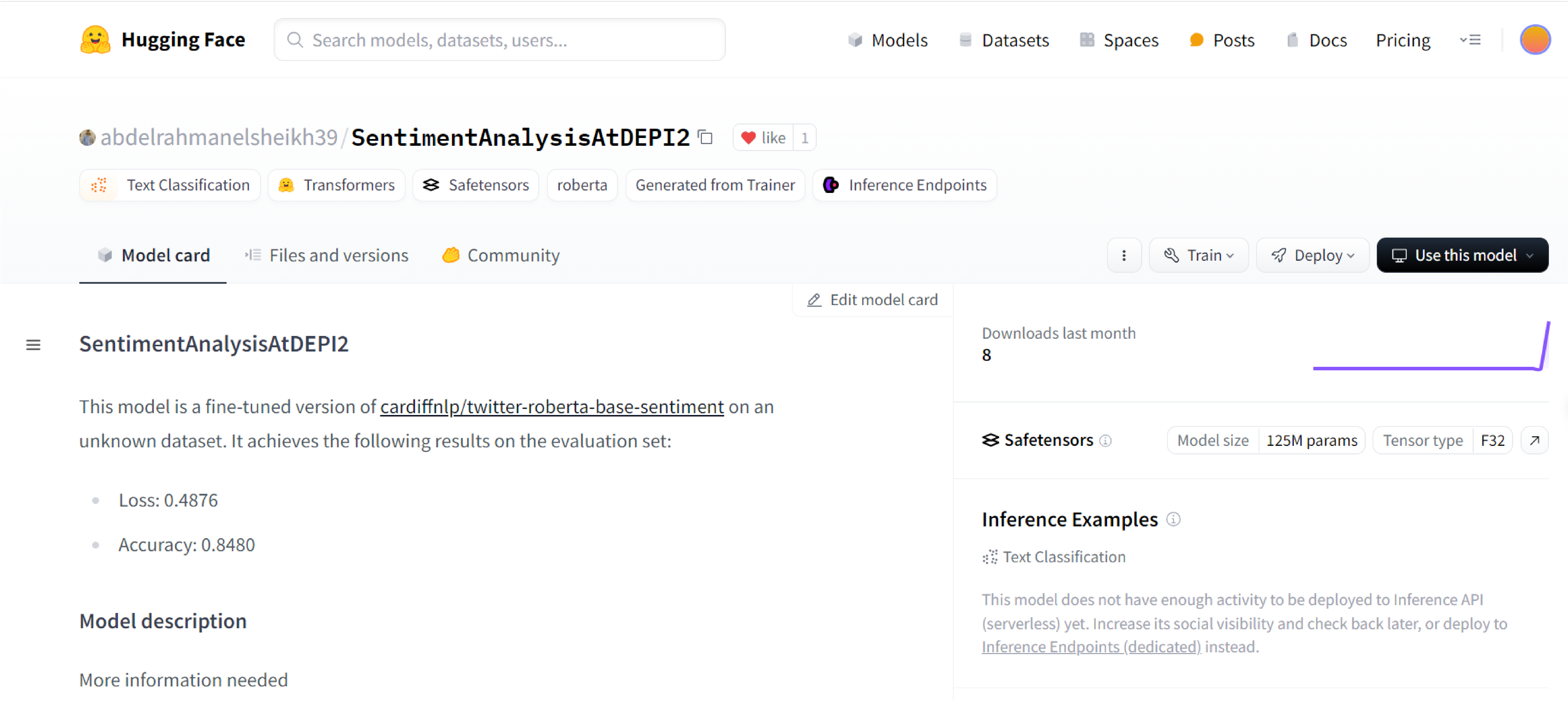


LSTM2 ROC CURVE

Hugging Face

we developed a sentiment analysis project using the CardiffNLP/twitter-roberta-base-sentiment model from Hugging Face to classify reviews into positive, negative, and neutral sentiments.

Our Model


Hugging Face
Search models, datasets, users...
Models
Datasets
Spaces
Posts
Docs
Pricing
abdelrahmanelsheikh39/SentimentAnalysisAtDEPI2
like 1
Text Classification
Transformers
Safetensors
roberta
Generated from Trainer
Inference Endpoints
Model card
Files and versions
Community
Train
Deploy
Use this model
Edit model card
SentimentAnalysisAtDEPI2
This model is a fine-tuned version of cardiffnlp/twitter-roberta-base-sentiment on an unknown dataset. It achieves the following results on the evaluation set:
Loss: 0.4876
Accuracy: 0.8480
Model description
More information needed
Downloads last month
8
Safetensors
Model size
125M params
Tensor type
F32
Inference Examples
Text Classification
This model does not have enough activity to be deployed to Inference API (serverless) yet. Increase its social visibility and check back later, or deploy to Inference Endpoints (dedicated) instead.

Training Result

Training Loss	Epoch	Step	Validation Loss	Accuracy
0.5493	1.0	14212	0.5040	0.8133
0.38	2.0	28424	0.4682	0.8371
0.3531	3.0	42636	0.4678	0.8433
0.3067	4.0	56848	0.4876	0.8480

The following hyperparameters were used during training:

- learning_rate: 2e-05
- train_batch_size: 32
- eval_batch_size: 32
- seed: 42
- optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- lr_scheduler_type: linear
- num_epochs: 4

Training Result

```
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("text-classification", model="cardiffnlp/twitter-roberta-base-sentiment")
pipe('I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product I
```

```
config.json:   0%|          | 0.00/747 [00:00<?, ?B/s]

pytorch_model.bin:   0%|          | 0.00/499M [00:00<?, ?B/s]

vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]

merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/150 [00:00<?, ?B/s]

Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` object. Mod
el will be on CPU.
```

```
text ="I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product
```

```
from transformers import pipeline
classifier = pipeline("text-classification", model="abdelrahmanelsheikh39/SentimentAnalysisAtDEPI2")
classifier(text)
```

```
config.json:   0%|          | 0.00/1.03k [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/499M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/1.24k [00:00<?, ?B/s]

vocab.json:   0%|          | 0.00/798k [00:00<?, ?B/s]

merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/3.56M [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/958 [00:00<?, ?B/s]

Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` object. Mod
el will be on CPU.

[{'label': 'LABEL_4', 'score': 0.8687160611152649}]
```

Training Result

Training results

Training Loss	Epoch	Step	Validation Loss	Accuracy
0.5583	1.0	14212	0.5413	0.7958
0.4939	2.0	28424	0.5007	0.8201
0.4564	3.0	42636	0.4923	0.8314

Training hyperparameters

The following hyperparameters were used during training:

- learning_rate: 2e-05
- train_batch_size: 32
- eval_batch_size: 32
- seed: 42
- optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- lr_scheduler_type: linear
- num_epochs: 3

Training **Result**

```
: # Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("text-classification", model="nlptown/bert-base-multilingual-uncased-sentiment")
pipe("I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product ]
```

```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` object. Mod
el will be on CPU.
```

```
: [{'label': '5 stars', 'score': 0.7185351848602295}]
```

```
text ="I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product
```

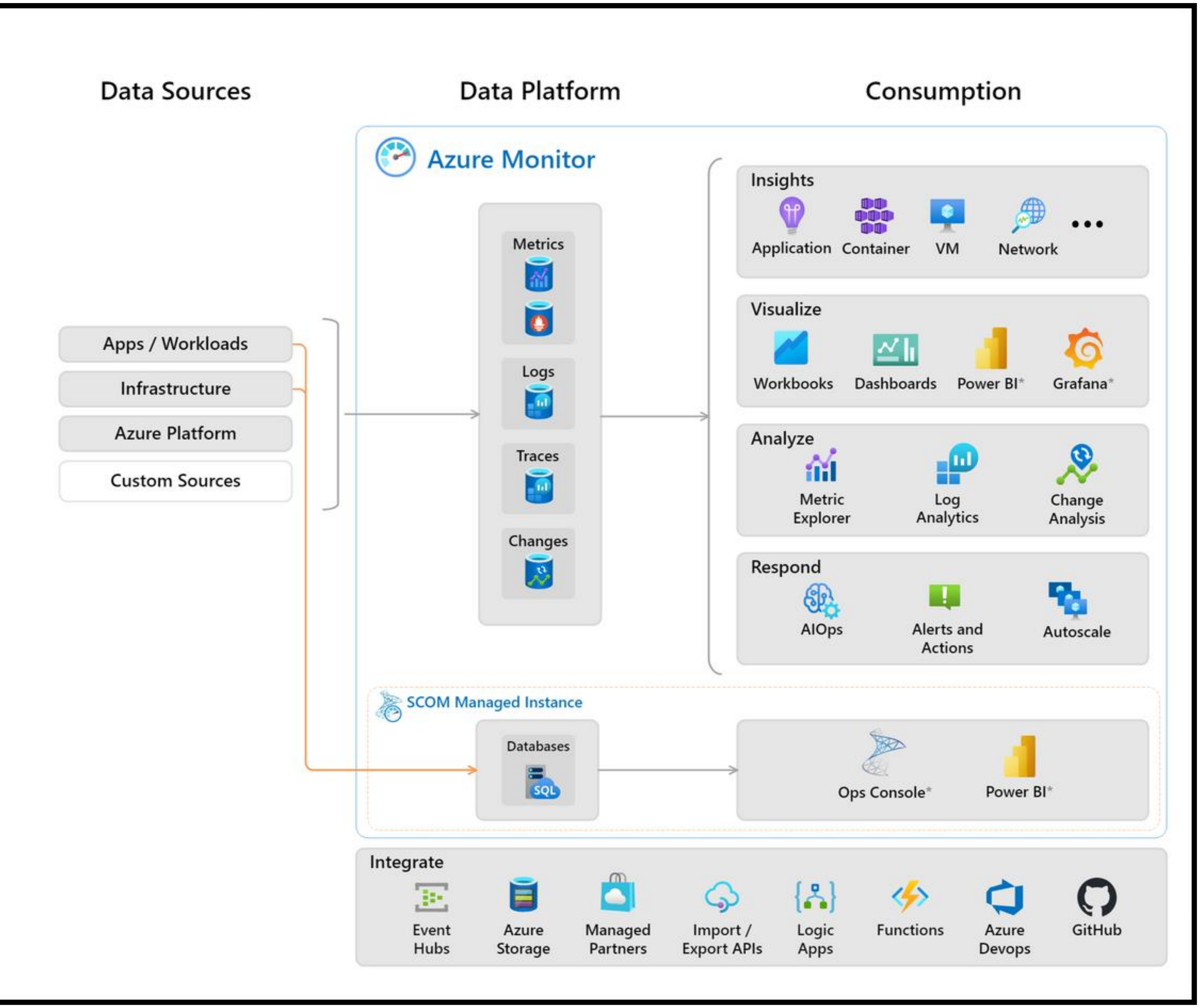
```
from transformers import pipeline
classifier = pipeline("text-classification", model="abdelrahmanelsheikh39/SentimentAnalysisAtDEPI")
classifier(text)
```

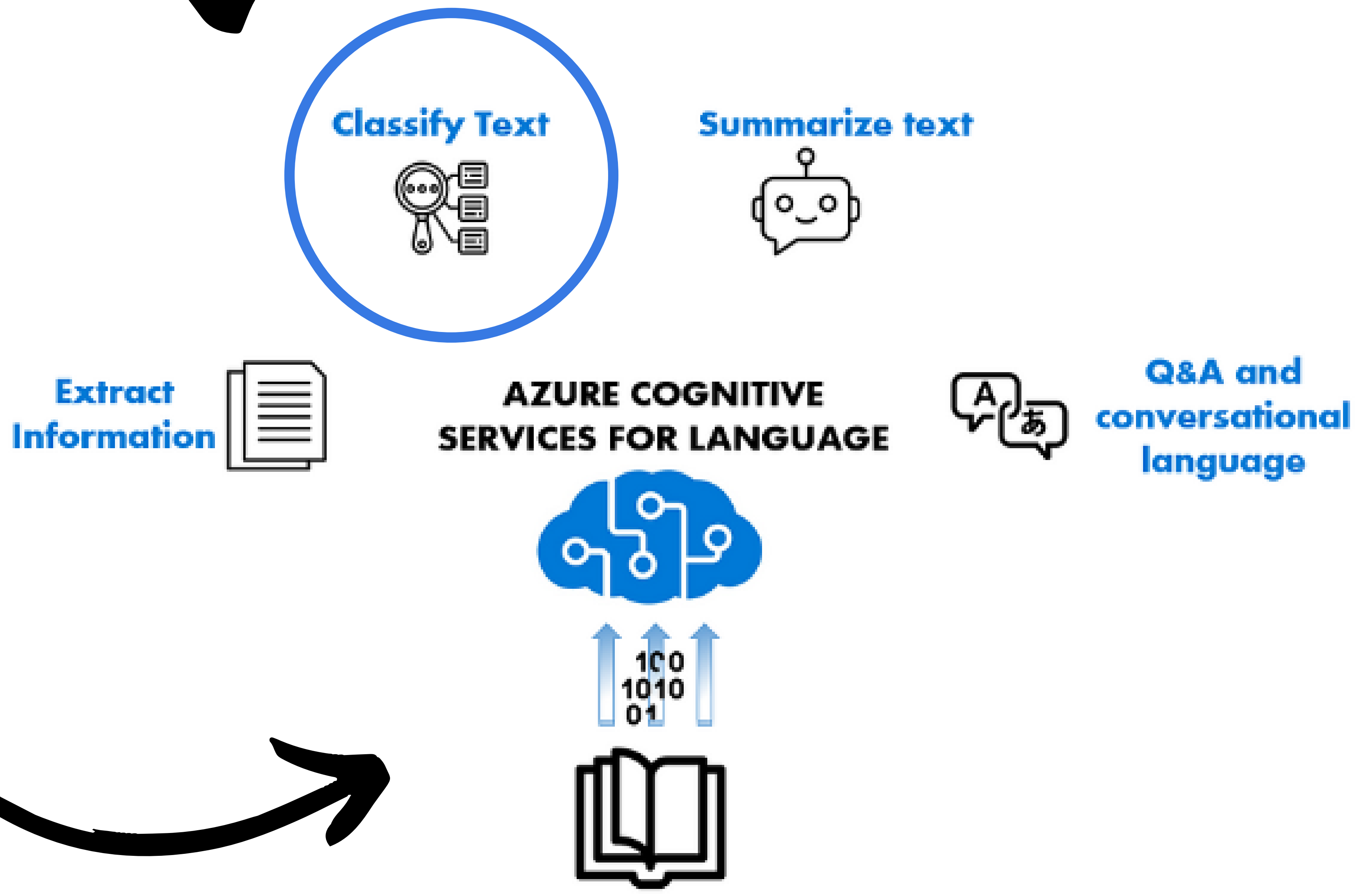
```
config.json:   0%|          | 0.00/1.23k [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/669M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/1.26k [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/872k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/2.56M [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/125 [00:00<?, ?B/s]
```

```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` object. Mod
el will be on CPU.
```

```
[{'label': '5 stars', 'score': 0.9597824215888977}]
```

Microsoft Azure





bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a
han a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.

example 1

positive review

Document sentiment

Positive

Confidence: 100.00%

100.00%	0.00%	0.00%
POSITIVE	NEUTRAL	NEGATIVE

I have bought several of the Vitality canned dog food products and have found them all to be of good quality.

Sentence sentiment

Positive

Confidence: 100.00%

100.00%	0.00%	0.00%
POSITIVE	NEUTRAL	NEGATIVE

The product looks more like a stew than a processed meat and it smells better.

Sentence sentiment

Positive

Confidence: 100.00%

100.00%	0.00%	0.00%
POSITIVE	NEUTRAL	NEGATIVE

If you're impulsive like me, then $6 is ok. Don't get me wrong, the quality of these babies is very good and I have no complaints. But in retrospect, the price is a little ridiculous (esp. when you add on the shipping).

Analyzed sentiment

Document sentiment

Mixed
Confidence: 57.00%

57.00%	9.00%	34.00%
POSITIVE	NEUTRAL	NEGATIVE

If you're impulsive like me, then $6 is ok.

Sentence sentiment

Positive
Confidence: 71.00%

71.00%	27.00%	2.00%
POSITIVE	NEUTRAL	NEGATIVE

assessment
assessment

Don't get me wrong, the quality of these babies is very good and I have no complaints.

target targ… as… assessme…

Sentence sentiment

Positive
Confidence: 100.00%

100.00%	0.00%	0.00%
POSITIVE	NEUTRAL	NEGATIVE

Opinion

Target: quality
Assessments:
good (positive, 100.00%)

Opinion

Target: babies
Assessments:
good (positive, 100.00%)

example 2

mixed review (mostly positive)

it was terrible experience

assessm...

it was terrible experience

assess... target

Sentence sentiment

Negative
Confidence: 0.00%

0.00%	0.00%	100.00%
POSITIVE	NEUTRAL	NEGATIVE

Opinion

Target: experience

Assessments:
terrible (negative, 99.00%)

Analyzed sentiment

Document sentiment

Negative
Confidence: 0.00%

0.00%	0.00%	100.00%
POSITIVE	NEUTRAL	NEGATIVE

example 3

Resource Page

scikit learn

TensorFlow

Hugging Face

kaggle

Thank You