ML1030 – Capstone Project
Assignment 4: Project Summary

Submitted by: Michael Vasiliou

Executive Summary

The application of machine learning in a clinical diagnostic setting opens boundless opportunities to improve patient care and health outcomes. The adoption of machine learning in this setting is often hampered by the lack of interpretability of results that occurs in some machine learning models. Ensuring and improving interpretability of results in a clinical setting would improve the ability of clinicians to understand the output of the machine learning model, and as a result improve the uptake of this supportive technology in their practice. To support this an interpretability framework was created. As the needs became clearer the project evolved to implementing a full machine learning pipeline.

Business Background

The overall research project goal is to improve the ability and acceptance of Machine Learning in clinical decision making. The research project is using multiple datasets including the IMDB movie review, and the MIMIC 3 clinical dataset. Through analyzing the disparate datasets, including the health care records and discharge summaries of 40,000 patients and comparing results and interpretability across multiple machine learning models the project aims to improve clinician trust, model interpretability, and uptake within a clinical setting

> *The output of the proposed system is an interpretable Knowledge Base, which can link the pattern groups, discovered characteristics of records, and patients' records together to shows "what" (disease), "who/where" (tracking patient records back) and "why" (discovered patterns) to interpret clinical notes for better clinical decision making.*
>
> - *Interpretability on Clinical Analysis from Pattern Disentanglement Insight (Appendix)*

Initially, the main goal of the sub-project I was focusing on was model interpretability by implementing post-hoc interpretability such as LIME, SHAP, Natural Indirect Effect (NIE), and Transformers Interpret. With the number of researchers, various data sets, and disparate models, it became clear that implementing a full Machine Learning pipeline would be extremely beneficial, promote consistency, transparency, and minimize coding time to allow researchers to focus on the results.

Solution Evolution

How it started

Initially, the project goal was to implement a solution to examine post-hoc model interpretability in a simple, straightforward manner. A post-hoc interpretability framework was created where researchers would plug in their trained model and datasets and allow them to generate interpretability results easily and quickly at both a global and local level using LIME and SHAP.

How it shifted

After the project ream reviewed the simplicity and benefits of the prototype framework it was decided that it would be beneficial to expand the scope to include not just the interpretability components but to build the entire ML pipeline.

Solution Design Requirements

There were three categories of requirements considered when designing the project and its solution: project requirements, scientific requirements, and technical requirements.

Project Requirements

The overall research project will extend well beyond the length of the ML1030 course. While I have committed to continue with the project beyond the end of the course, the longevity and usability of the final product will rely heavily on:

- Transfer of knowledge
- Maintainability

Scientific Requirements

The results from a scientific research project must adhere to some stringent requirements. Within the project itself the process, must be repeatable and comparable between researchers on the project. Additionally, the process must be documented and transparent enough to withstand scrutiny and repeatability from peers reviewing the papers. Four key requirements were highlighted in the design of this project to ensure this was possible:

- Repeatability
- Consistency
- Comparability
- Documentation

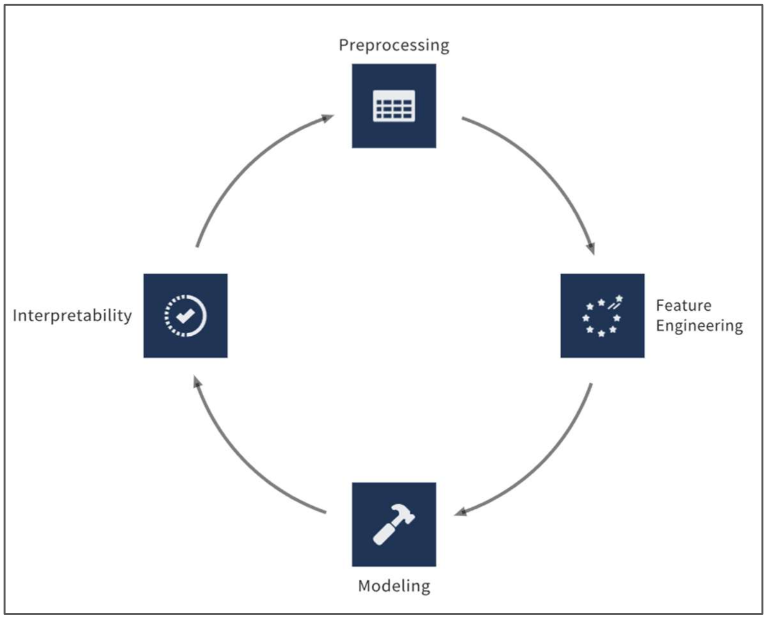
Technical Requirements

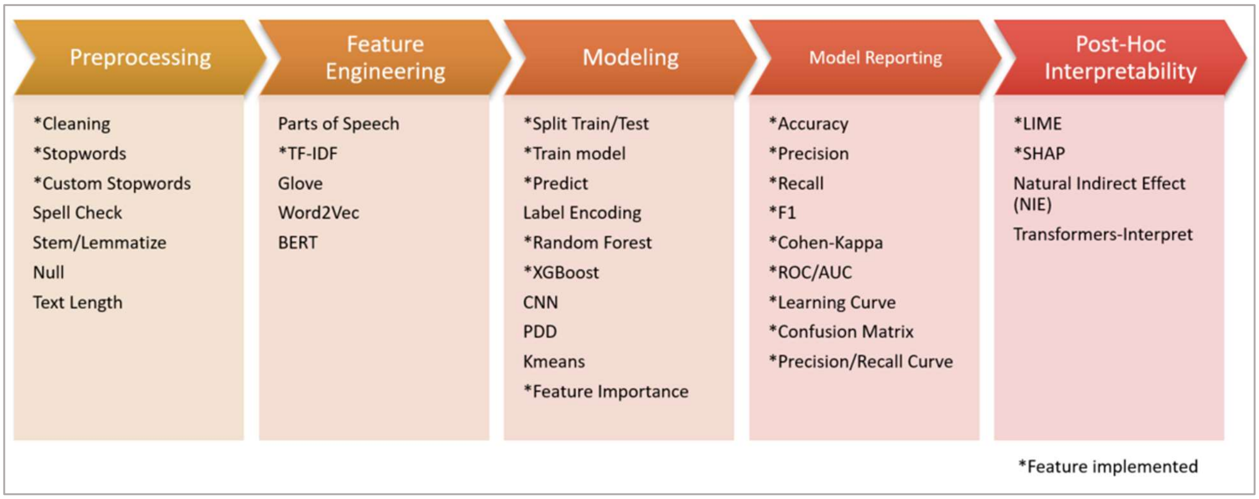
At this stage only high-level technical requirements are available. The detailed use cases and lower-level requirements will evolve as the project moves forward and the researchers are able to integrate the pipeline into their activities. The beginning technical requirements are:

- Ease of use
- Ease of reporting/analysis
- Low implementation overhead
- Use on multiple platforms (not required to included distributed at this point)
- Models used (KMeans, CNN, Random Forest, XGBoost, PDD)
- Interpretability (e.g., LIME, SHAP, Natural Indirect Effect (NIE), Transformers-Interpret)
- Model Reporting (e.g., Accuracy, Precision, learning curve, ROC/AUC)

Solution Overview

The pipeline being developed is separated into 4 main stages: preprocessing, feature engineering, modeling, and interpretability/analysis. At this stage, the pipeline is operational and usable from end to end, however not all identified functionality has been implemented. Additional functionality requirements will be included as the project continues to move forward.





Preprocessing	Feature Engineering	Modeling	Model Reporting	Post-Hoc Interpretability
*Cleaning	Parts of Speech	*Split Train/Test	*Accuracy	*LIME
*Stopwords	*TF-IDF	*Train model	*Precision	*SHAP
*Custom Stopwords	Glove	*Predict	*Recall	Natural Indirect Effect (NIE)
Spell Check	Word2Vec	Label Encoding	*F1	Transformers-Interpret
Stem/Lemmatize	BERT	*Random Forest	*Cohen-Kappa	
Null		*XGBoost	*ROC/AUC	
Text Length		CNN	*Learning Curve	
		PDD	*Confusion Matrix	
		Kmeans	*Precision/Recall Curve	
		*Feature Importance		

*Feature implemented

*Modeling currently only designed for supervised. Still need to add clustering.

Pipeline usage

To demonstrate the current functionality of the pipeline we will walk through a simple scenario utilizing the IMDB movie review database and comparing two models: XGBoost, and Random Forest. There are three areas that need to be setup prior to processing the pipeline: data, pipeline parameters, adding models.

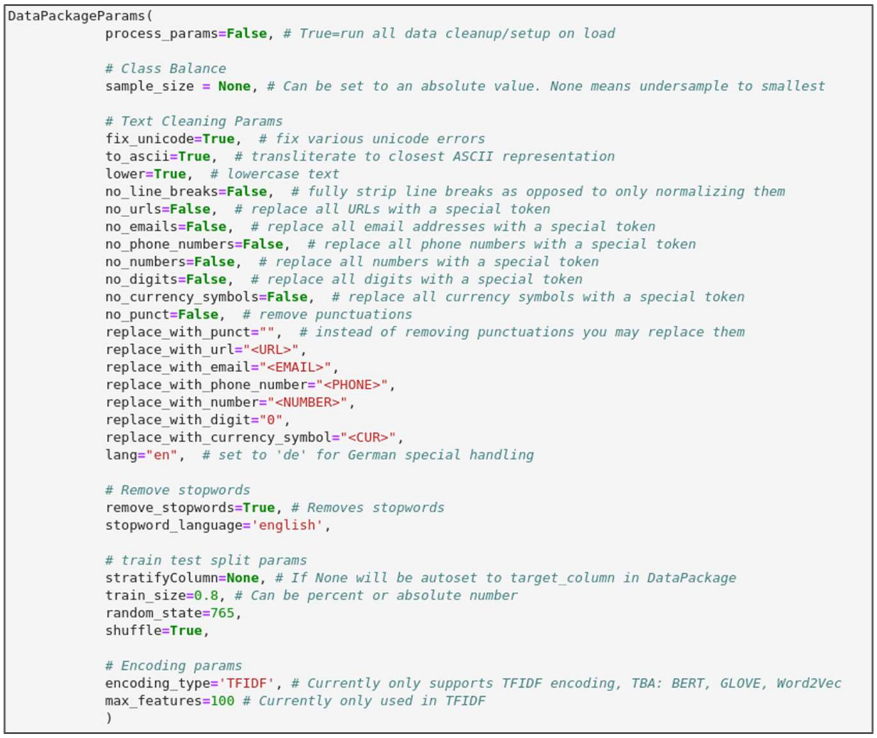
Data setup

The IMDb movie review database is a simple two column database with the review content in one column and the labelled sentiment of the review in the second column. The data setup only requires a few simple steps:

1. Load the dataset into a Pandas dataframe
2. Identify the target column
3. Identify the data column
4. Identify the unique column (if none, one will be added in the pipeline)

Parameter setup

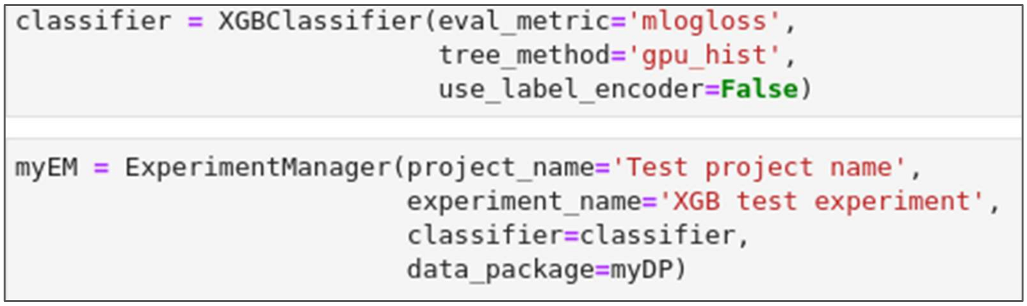
There are currently twenty-nine parameters that can be configured to customize the pipeline processing and are stored together in the DataPackageParams class. Each of these parameters has a default value set but can also be overridden by changing it in the DataPackageParams class.



```
DataPackageParams(
            process_params=False, # True=run all data cleanup/setup on load

            # Class Balance
            sample_size = None, # Can be set to an absolute value. None means undersample to smallest

            # Text Cleaning Params
            fix_unicode=True,  # fix various unicode errors
            to_ascii=True,  # transliterate to closest ASCII representation
            lower=True,  # lowercase text
            no_line_breaks=False,  # fully strip line breaks as opposed to only normalizing them
            no_urls=False,  # replace all URLs with a special token
            no_emails=False,  # replace all email addresses with a special token
            no_phone_numbers=False,  # replace all phone numbers with a special token
            no_numbers=False,  # replace all numbers with a special token
            no_digits=False,  # replace all digits with a special token
            no_currency_symbols=False,  # replace all currency symbols with a special token
            no_punct=False,  # remove punctuations
            replace_with_punct="",  # instead of removing punctuations you may replace them
            replace_with_url="<URL>",
            replace_with_email="<EMAIL>",
            replace_with_phone_number="<PHONE>",
            replace_with_number="<NUMBER>",
            replace_with_digit="0",
            replace_with_currency_symbol="<CUR>",
            lang="en",  # set to 'de' for German special handling

            # Remove stopwords
            remove_stopwords=True, # Removes stopwords
            stopword_language='english',

            # train test split params
            stratifyColumn=None, # If None will be autoset to target_column in DataPackage
            train_size=0.8, # Can be percent or absolute number
            random_state=765,
            shuffle=True,

            # Encoding params
            encoding_type='TFIDF', # Currently only supports TFIDF encoding, TBA: BERT, GLOVE, Word2Vec
            max_features=100 # Currently only used in TFIDF
            )
```

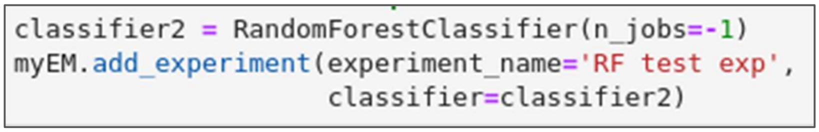
Model setup

On initiation of the pipeline, governed by the ExperimentManager class we add our initial model, data, and parameters.



```
classifier = XGBClassifier(eval_metric='mlogloss',
                           tree_method='gpu_hist',
                           use_label_encoder=False)

myEM = ExperimentManager(project_name='Test project name',
                         experiment_name='XGB test experiment',
                         classifier=classifier,
                         data_package=myDP)
```

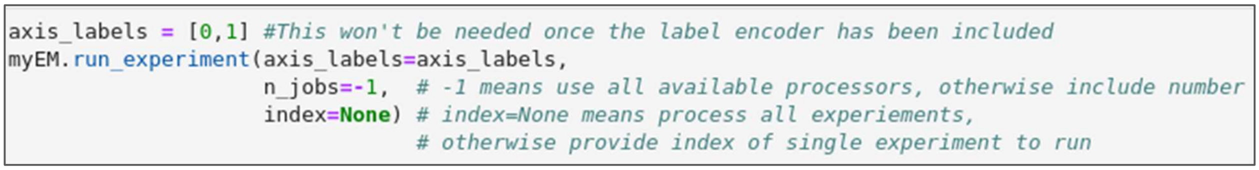
Then our second model is added as a new experiment. Further models can be added as desired.



```
classifier2 = RandomForestClassifier(n_jobs=-1)
myEM.add_experiment(experiment_name='RF test exp',
                    classifier=classifier2)
```

Run Experiments

Once the data, parameters, and model are loaded into the pipeline the only thing left to do is “run experiment.” This will process the data package for cleaning, split, balance, train the models, process the learning curves, and generate any other statistics required for reporting.



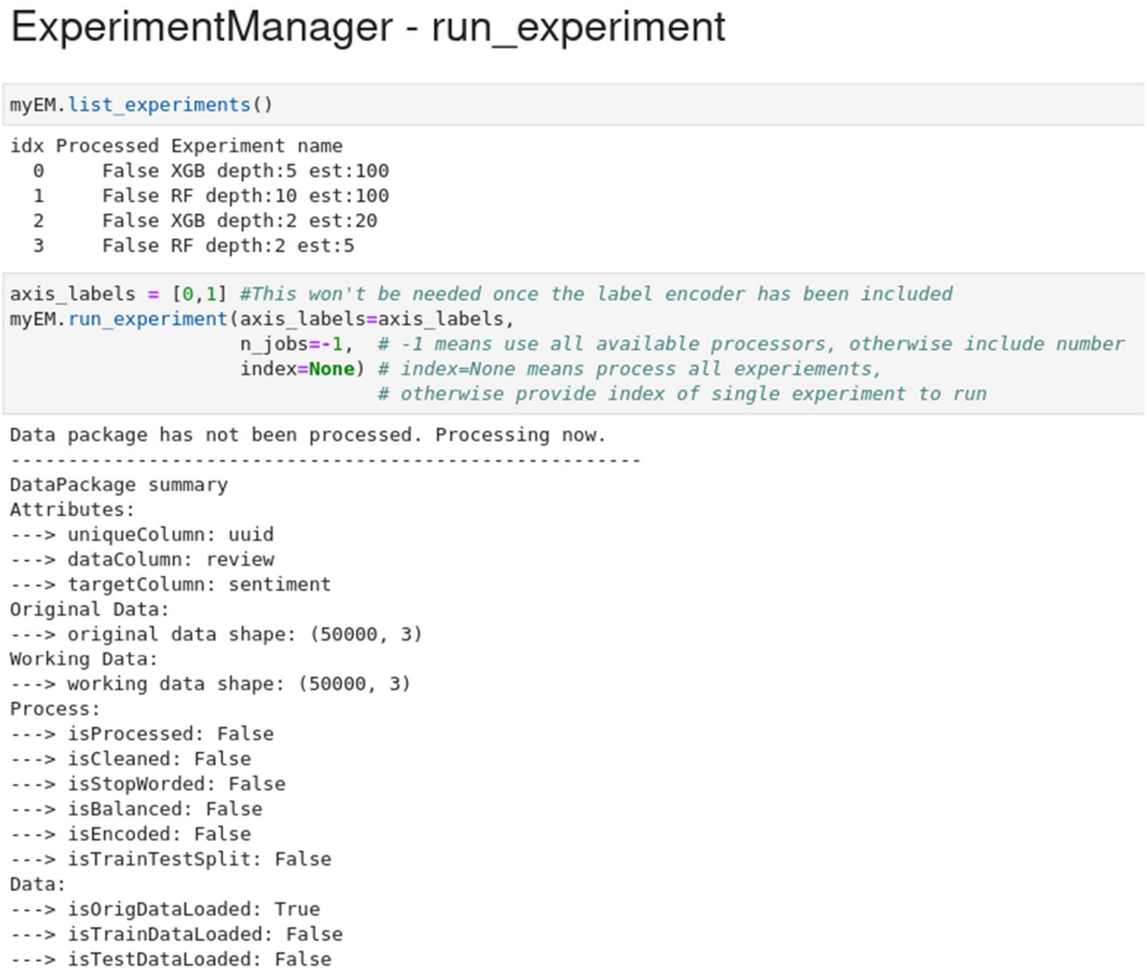
```
axis_labels = [0,1] #This won't be needed once the label encoder has been included
myEM.run_experiment(axis_labels=axis_labels,
                    n_jobs=-1,  # -1 means use all available processors, otherwise include number
                    index=None) # index=None means process all experiements,
                                # otherwise provide index of single experiment to run
```

**Please see Appendix A for the pipeline processing output.
**Please see Appendix B for the example output from a single model

Conclusion and Next Steps

The skeleton of the pipeline framework has been successfully developed and meets the project, scientific, and design requirements set out at the start. To support knowledge transfer and project growth, three team members will be collaborating with me to implement additional features, document the solution, and assist other research project team members in implementing the solution within their own daily work.

Appendix A – Pipeline Processing Output



ExperimentManager - run_experiment

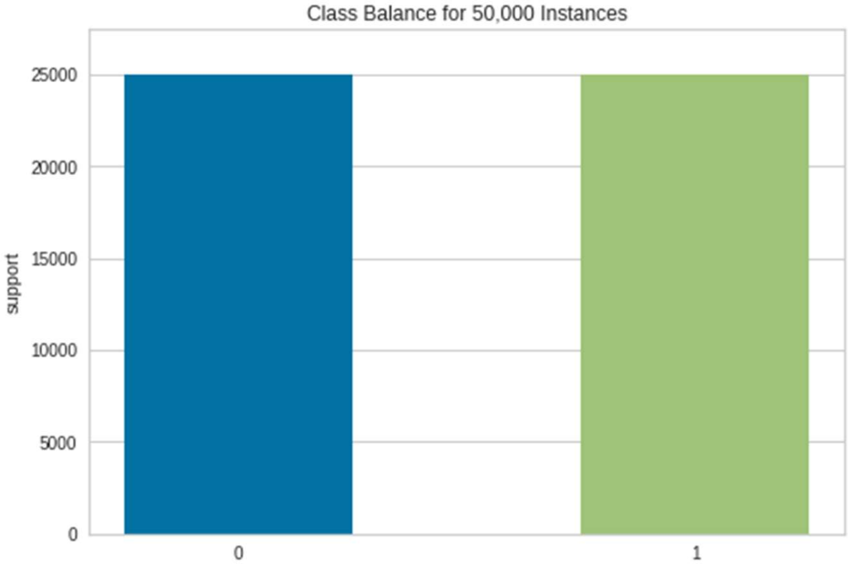
```
myEM.list_experiments()
```

```
idx Processed Experiment name
  0     False XGB depth:5 est:100
  1     False RF depth:10 est:100
  2     False XGB depth:2 est:20
  3     False RF depth:2 est:5
```

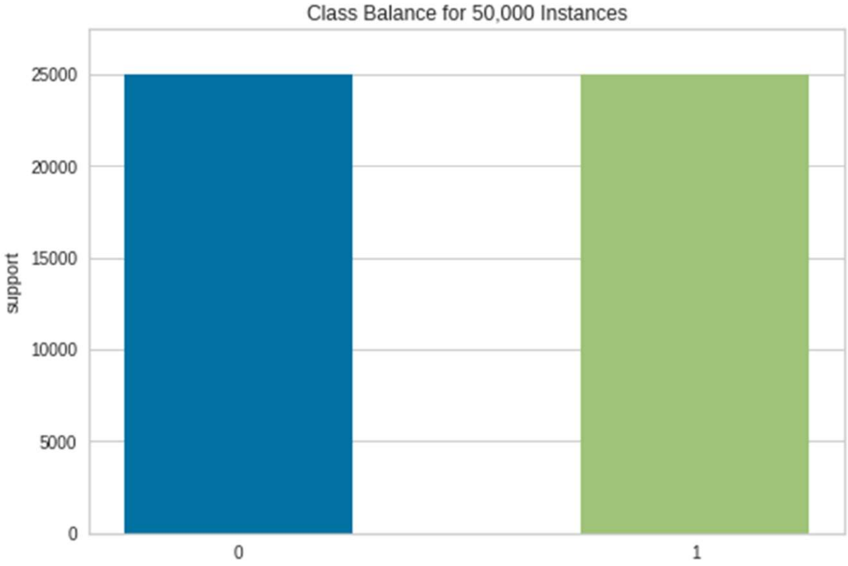
```
axis_labels = [0,1] #This won't be needed once the label encoder has been included
myEM.run_experiment(axis_labels=axis_labels,
                    n_jobs=-1,  # -1 means use all available processors, otherwise include number
                    index=None) # index=None means process all experiements,
                                # otherwise provide index of single experiment to run
```

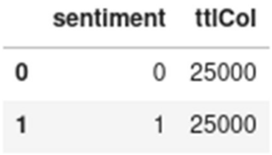
```
Data package has not been processed. Processing now.
-------------------------------------------------------
DataPackage summary
Attributes:
---> uniqueColumn: uuid
---> dataColumn: review
---> targetColumn: sentiment
Original Data:
---> original data shape: (50000, 3)
Working Data:
---> working data shape: (50000, 3)
Process:
---> isProcessed: False
---> isCleaned: False
---> isStopWorded: False
---> isBalanced: False
---> isEncoded: False
---> isTrainTestSplit: False
Data:
---> isOrigDataLoaded: True
---> isTrainDataLoaded: False
---> isTestDataLoaded: False
```

```
Processing data package with provided parameters
```

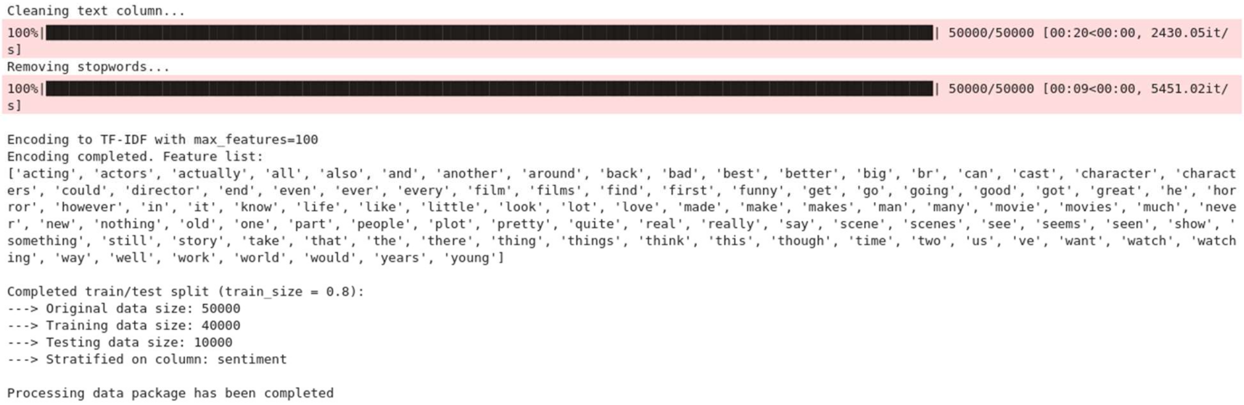


```
Undersampling data to match min class: 0 of size: 25000
```

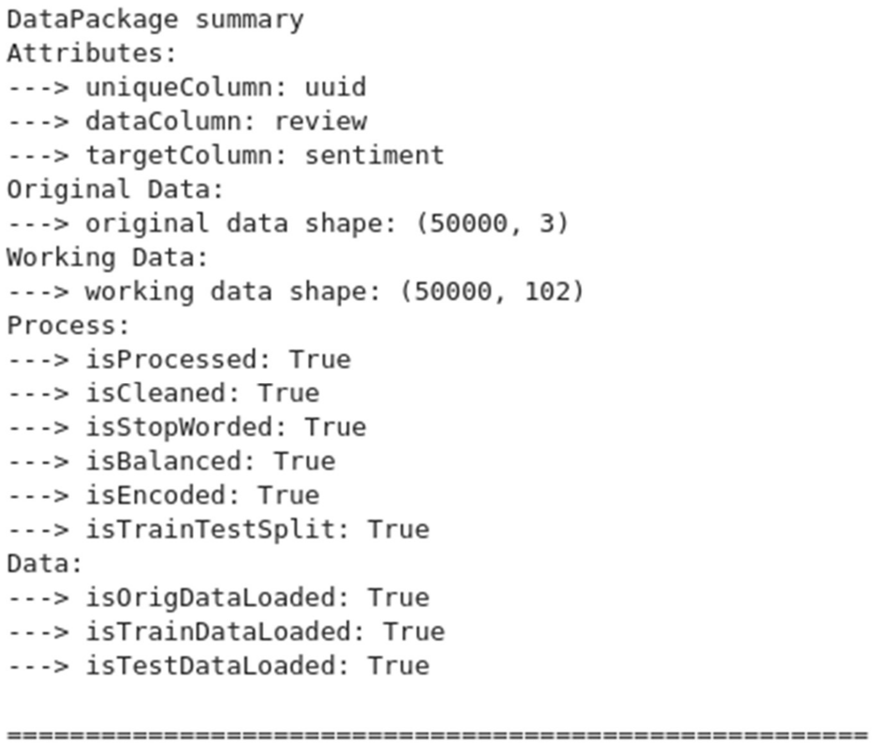




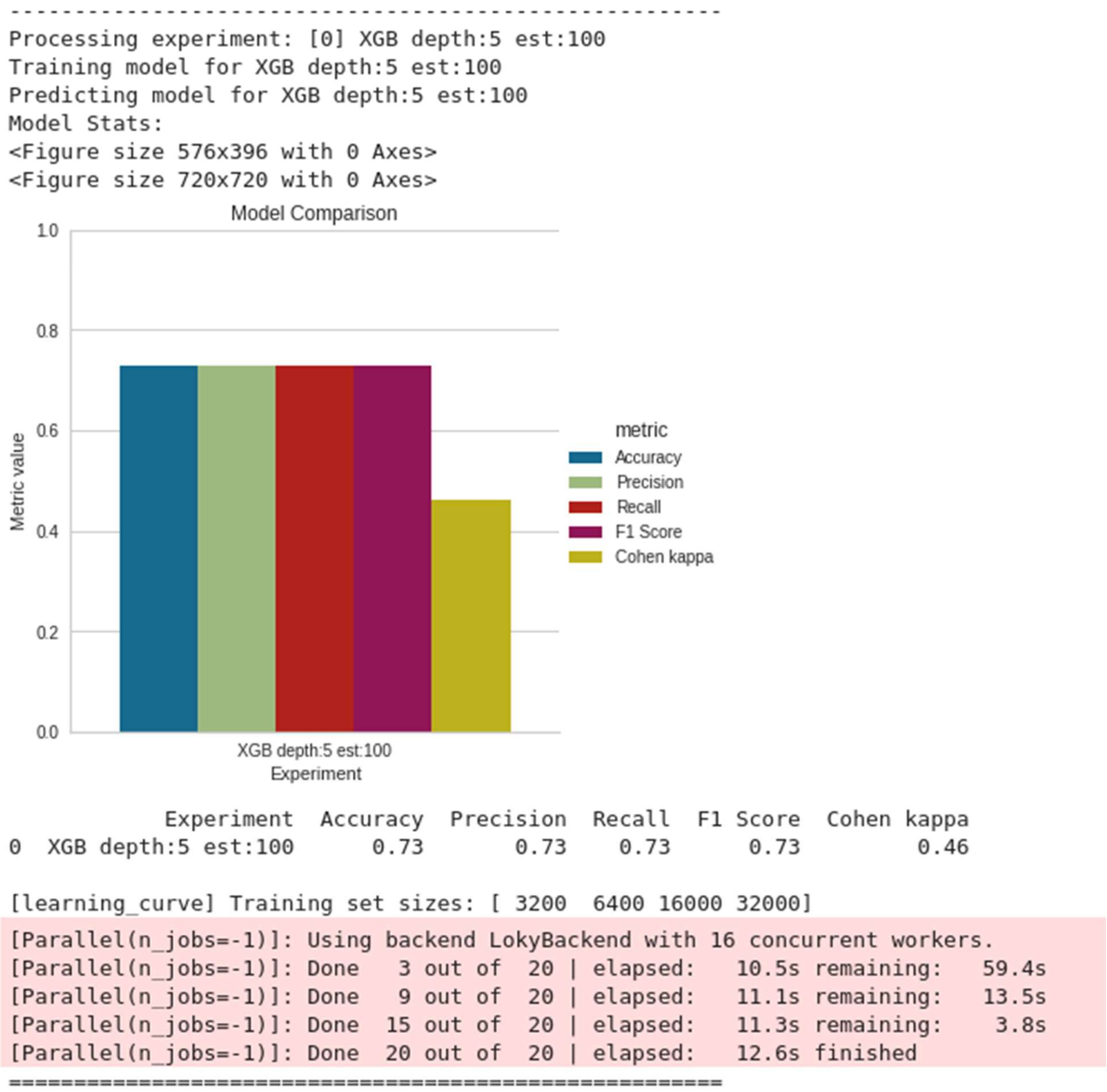
	sentiment	ttlCol
0	0	25000
1	1	25000



```
Cleaning text column...
100%|██████████████████████████████████████████████████████████████████████████████████████████| 50000/50000 [00:20<00:00, 2430.05it/
s]
Removing stopwords...
100%|██████████████████████████████████████████████████████████████████████████████████████████| 50000/50000 [00:09<00:00, 5451.02it/
s]

Encoding to TF-IDF with max_features=100
Encoding completed. Feature list:
['acting', 'actors', 'actually', 'all', 'also', 'and', 'another', 'around', 'back', 'bad', 'best', 'better', 'big', 'br', 'can', 'cast', 'character', 'charact
ers', 'could', 'director', 'end', 'even', 'ever', 'every', 'film', 'films', 'find', 'first', 'funny', 'get', 'go', 'going', 'good', 'got', 'great', 'he', 'hor
ror', 'however', 'in', 'it', 'know', 'life', 'like', 'little', 'look', 'lot', 'love', 'made', 'make', 'makes', 'man', 'many', 'movie', 'movies', 'much', 'neve
r', 'new', 'nothing', 'old', 'one', 'part', 'people', 'plot', 'pretty', 'quite', 'real', 'really', 'say', 'scene', 'scenes', 'see', 'seems', 'seen', 'show', '
something', 'still', 'story', 'take', 'that', 'the', 'there', 'thing', 'things', 'think', 'this', 'though', 'time', 'two', 'us', 've', 'want', 'watch', 'watch
ing', 'way', 'well', 'work', 'world', 'would', 'years', 'young']

Completed train/test split (train_size = 0.8):
---> Original data size: 50000
---> Training data size: 40000
---> Testing data size: 10000
---> Stratified on column: sentiment

Processing data package has been completed
```

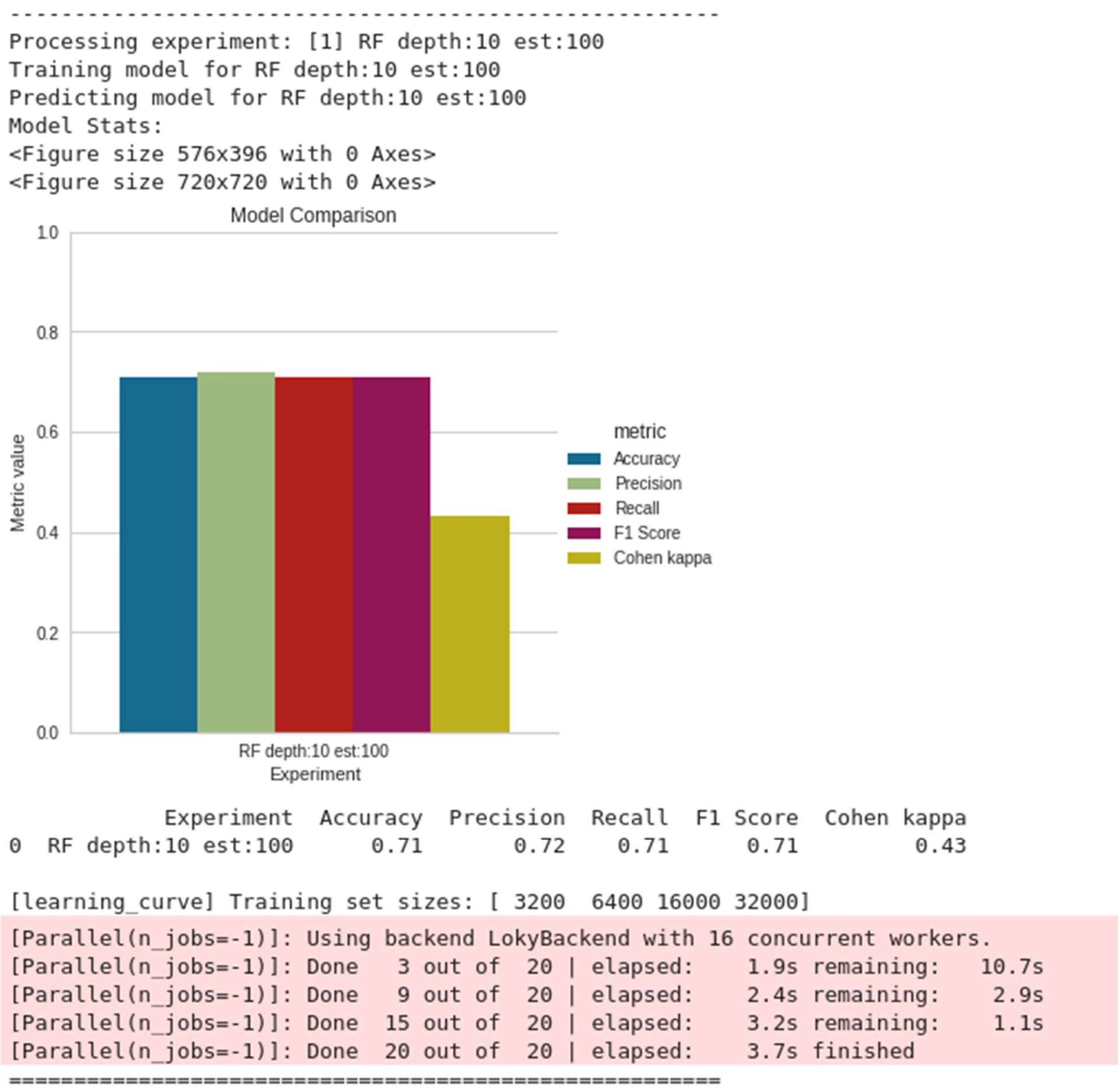


```
DataPackage summary
Attributes:
---> uniqueColumn: uuid
---> dataColumn: review
---> targetColumn: sentiment
Original Data:
---> original data shape: (50000, 3)
Working Data:
---> working data shape: (50000, 102)
Process:
---> isProcessed: True
---> isCleaned: True
---> isStopWorded: True
---> isBalanced: True
---> isEncoded: True
---> isTrainTestSplit: True
Data:
---> isOrigDataLoaded: True
---> isTrainDataLoaded: True
---> isTestDataLoaded: True

=======================================================
```



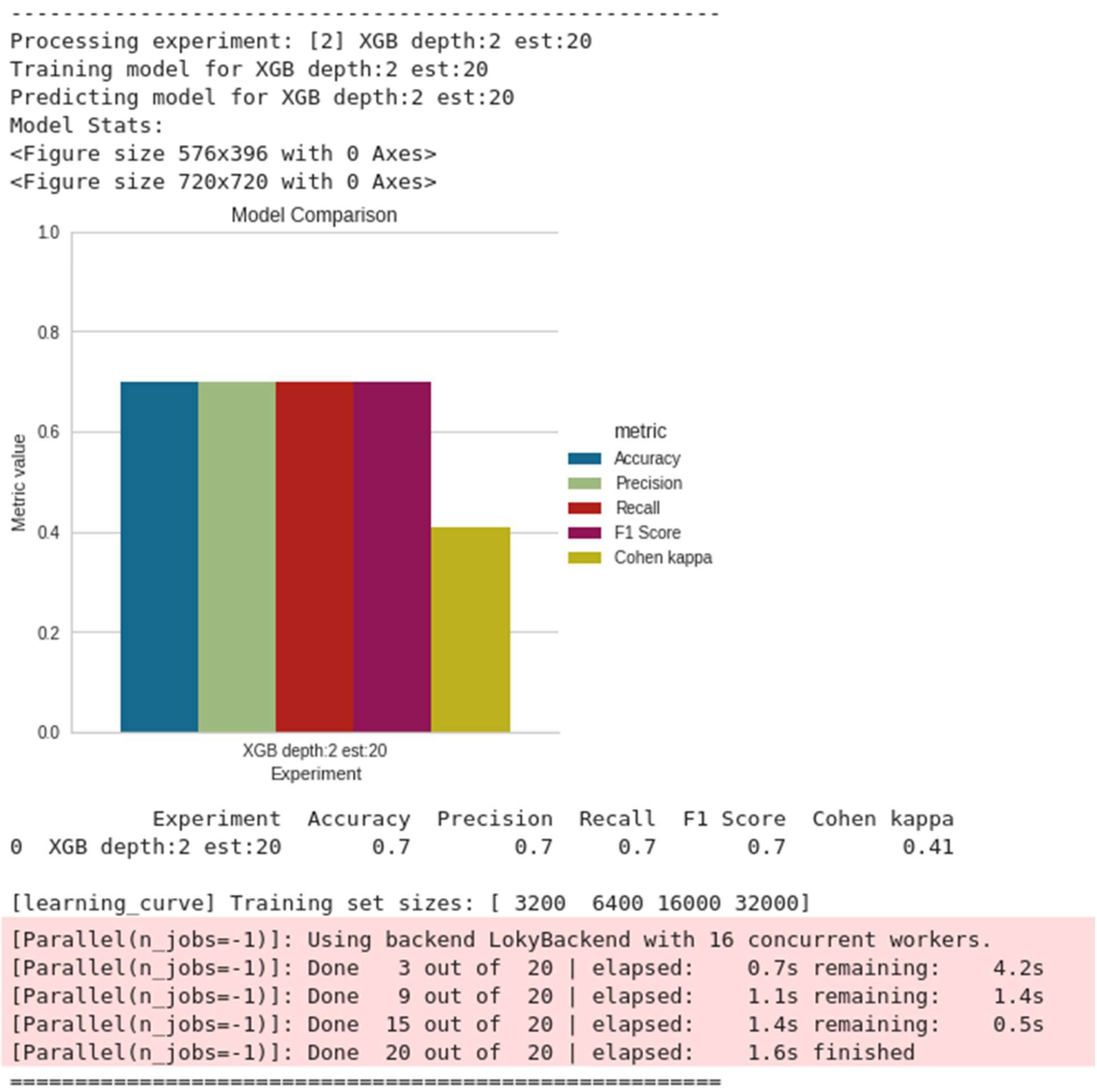
```
-------------------------------------------------------
Processing experiment: [0] XGB depth:5 est:100
Training model for XGB depth:5 est:100
Predicting model for XGB depth:5 est:100
Model Stats:
<Figure size 576x396 with 0 Axes>
<Figure size 720x720 with 0 Axes>
```

```
            Experiment  Accuracy  Precision  Recall  F1 Score  Cohen kappa
0  XGB depth:5 est:100      0.73       0.73    0.73      0.73         0.46

[learning_curve] Training set sizes: [ 3200  6400 16000 32000]
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  20 | elapsed:   10.5s remaining:   59.4s
[Parallel(n_jobs=-1)]: Done   9 out of  20 | elapsed:   11.1s remaining:   13.5s
[Parallel(n_jobs=-1)]: Done  15 out of  20 | elapsed:   11.3s remaining:    3.8s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:   12.6s finished
=======================================================
```



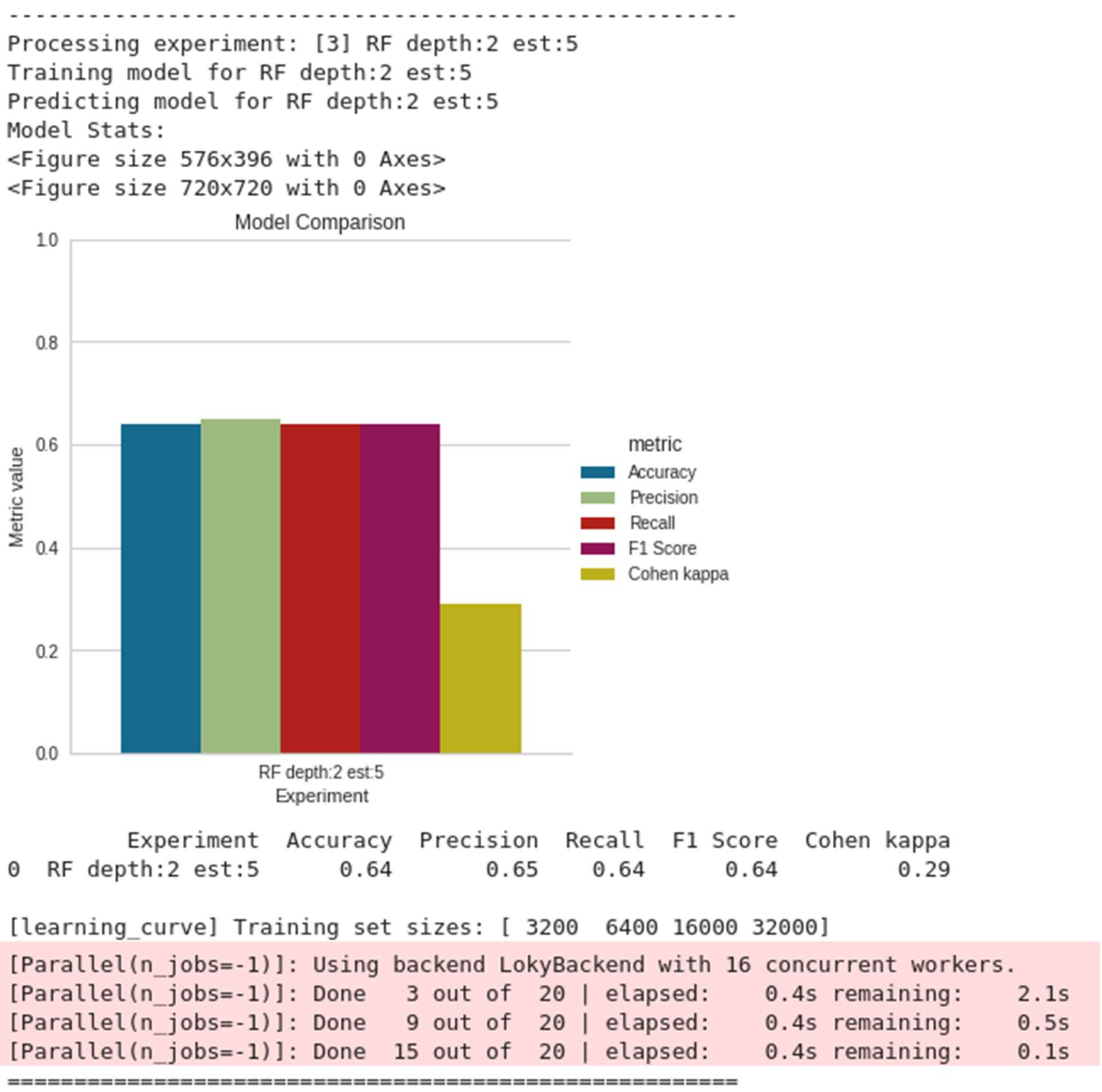
```
-------------------------------------------------------
Processing experiment: [1] RF depth:10 est:100
Training model for RF depth:10 est:100
Predicting model for RF depth:10 est:100
Model Stats:
<Figure size 576x396 with 0 Axes>
<Figure size 720x720 with 0 Axes>
```

```
            Experiment  Accuracy  Precision  Recall  F1 Score  Cohen kappa
0  RF depth:10 est:100      0.71       0.72    0.71      0.71         0.43

[learning_curve] Training set sizes: [ 3200  6400 16000 32000]
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  20 | elapsed:    1.9s remaining:   10.7s
[Parallel(n_jobs=-1)]: Done   9 out of  20 | elapsed:    2.4s remaining:    2.9s
[Parallel(n_jobs=-1)]: Done  15 out of  20 | elapsed:    3.2s remaining:    1.1s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    3.7s finished
=======================================================
```



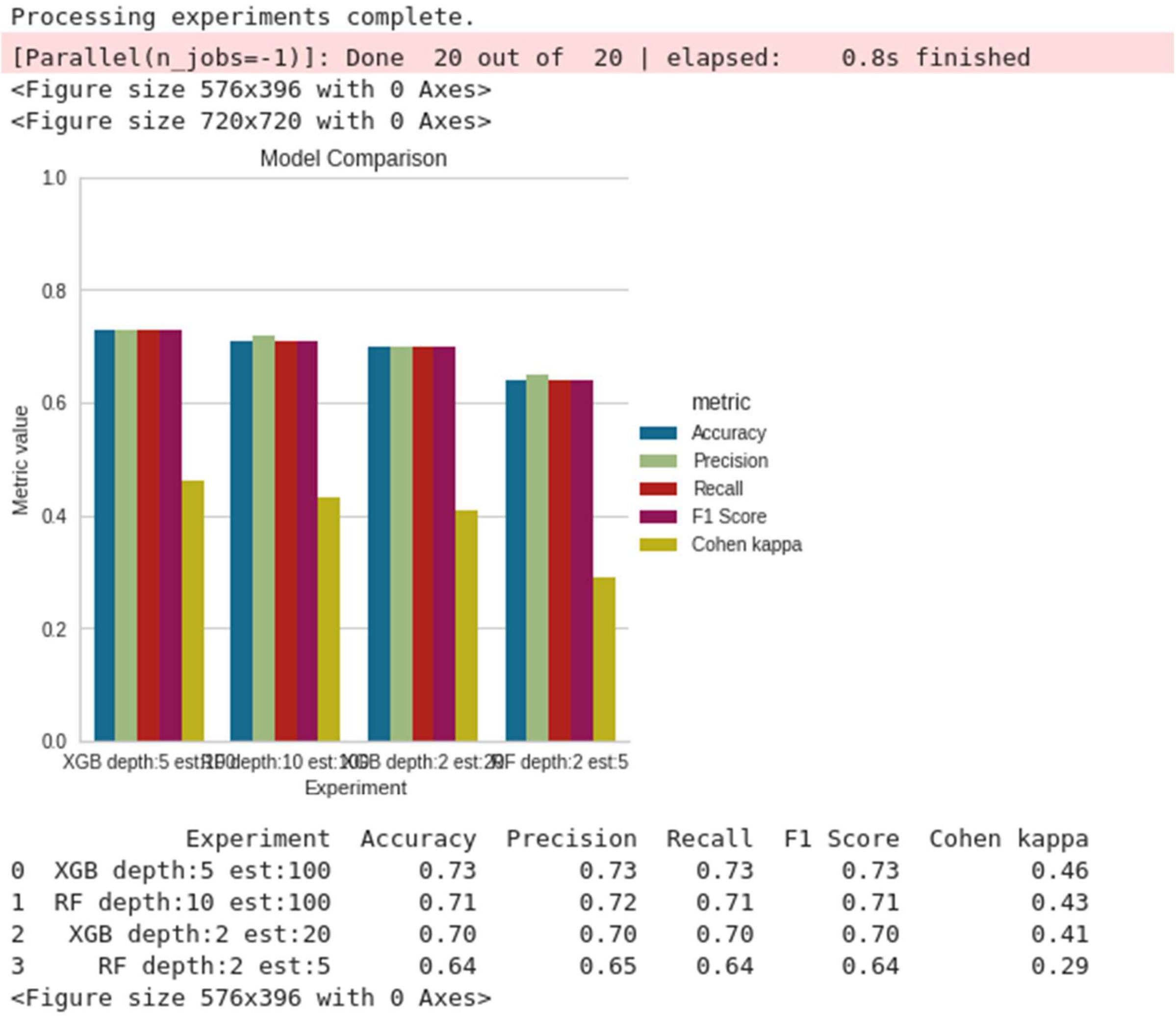
```
-------------------------------------------------------
Processing experiment: [2] XGB depth:2 est:20
Training model for XGB depth:2 est:20
Predicting model for XGB depth:2 est:20
Model Stats:
<Figure size 576x396 with 0 Axes>
<Figure size 720x720 with 0 Axes>
```

```
           Experiment  Accuracy  Precision  Recall  F1 Score  Cohen kappa
0  XGB depth:2 est:20       0.7        0.7     0.7       0.7         0.41

[learning_curve] Training set sizes: [ 3200  6400 16000 32000]
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  20 | elapsed:    0.7s remaining:    4.2s
[Parallel(n_jobs=-1)]: Done   9 out of  20 | elapsed:    1.1s remaining:    1.4s
[Parallel(n_jobs=-1)]: Done  15 out of  20 | elapsed:    1.4s remaining:    0.5s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    1.6s finished
=======================================================
```



```
-------------------------------------------------------
Processing experiment: [3] RF depth:2 est:5
Training model for RF depth:2 est:5
Predicting model for RF depth:2 est:5
Model Stats:
<Figure size 576x396 with 0 Axes>
<Figure size 720x720 with 0 Axes>
```

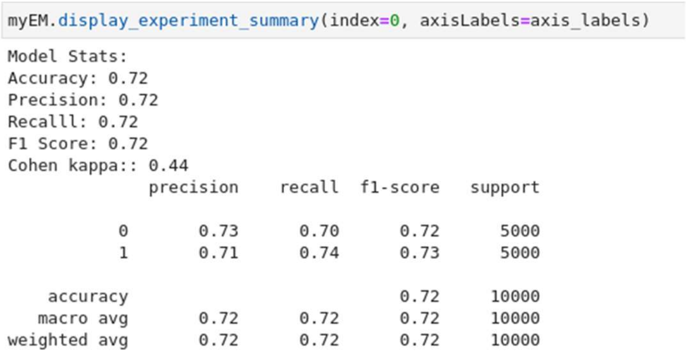
```
         Experiment  Accuracy  Precision  Recall  F1 Score  Cohen kappa
0  RF depth:2 est:5      0.64       0.65    0.64      0.64         0.29

[learning_curve] Training set sizes: [ 3200  6400 16000 32000]
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  20 | elapsed:    0.4s remaining:    2.1s
[Parallel(n_jobs=-1)]: Done   9 out of  20 | elapsed:    0.4s remaining:    0.5s
[Parallel(n_jobs=-1)]: Done  15 out of  20 | elapsed:    0.4s remaining:    0.1s
=======================================================
```



```
Processing experiments complete.
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    0.8s finished
<Figure size 576x396 with 0 Axes>
<Figure size 720x720 with 0 Axes>
```

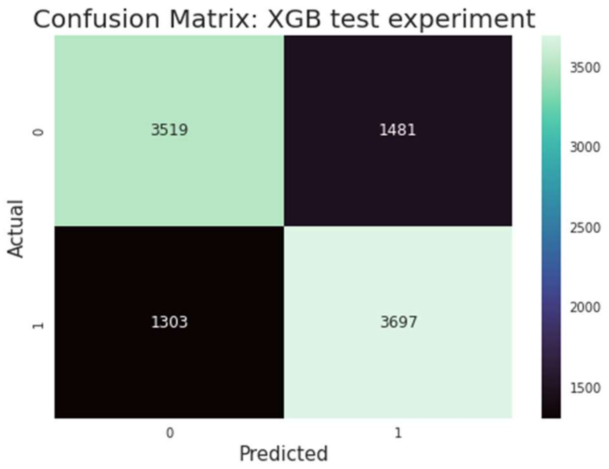
```
            Experiment  Accuracy  Precision  Recall  F1 Score  Cohen kappa
0  XGB depth:5 est:100      0.73       0.73    0.73      0.73         0.46
1  RF depth:10 est:100      0.71       0.72    0.71      0.71         0.43
2   XGB depth:2 est:20      0.70       0.70    0.70      0.70         0.41
3     RF depth:2 est:5      0.64       0.65    0.64      0.64         0.29
<Figure size 576x396 with 0 Axes>
```

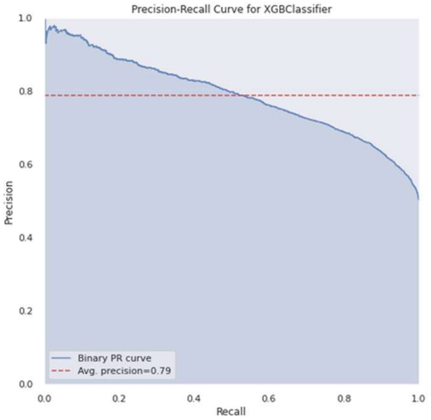
Appendix B – Single Model Output



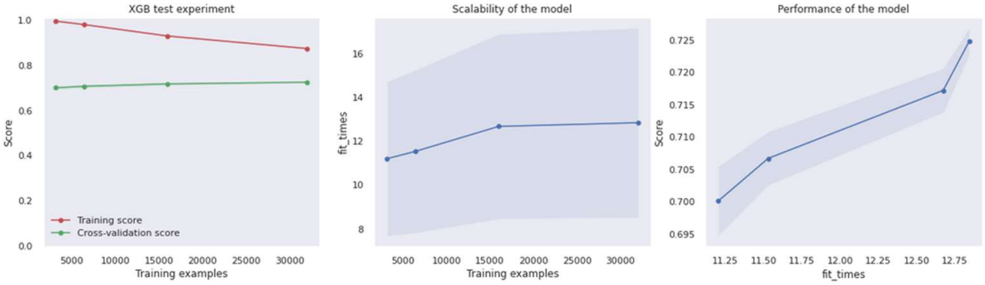
```
myEM.display_experiment_summary(index=0, axisLabels=axis_labels)
```

```
Model Stats:
Accuracy: 0.72
Precision: 0.72
Recalll: 0.72
F1 Score: 0.72
Cohen kappa:: 0.44
              precision    recall  f1-score   support

           0       0.73      0.70      0.72      5000
           1       0.71      0.74      0.73      5000

    accuracy                           0.72     10000
   macro avg       0.72      0.72      0.72     10000
weighted avg       0.72      0.72      0.72     10000
```

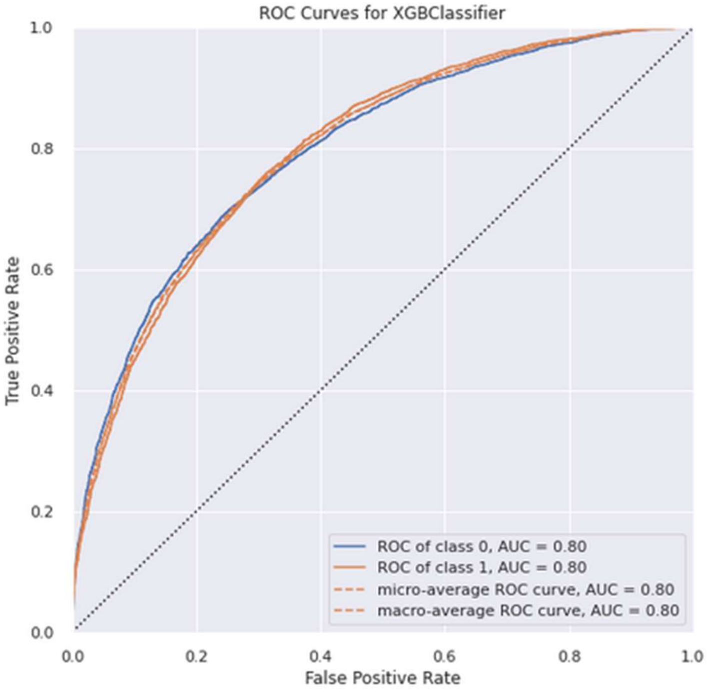




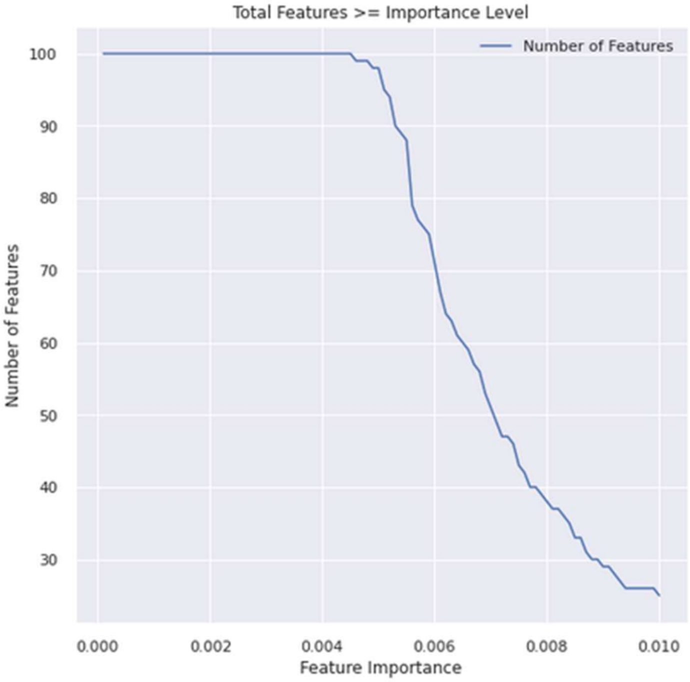
```
<Figure size 576x576 with 0 Axes>
```

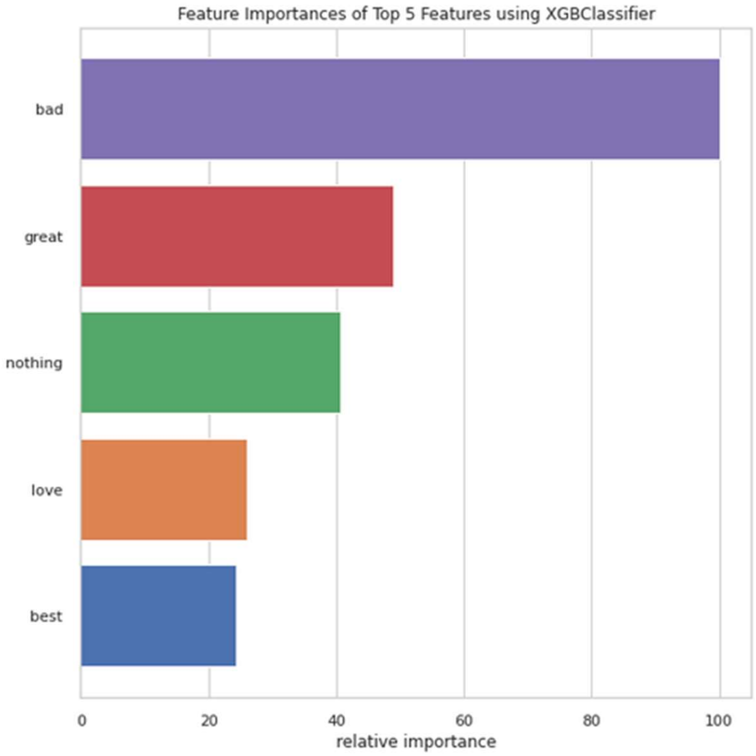


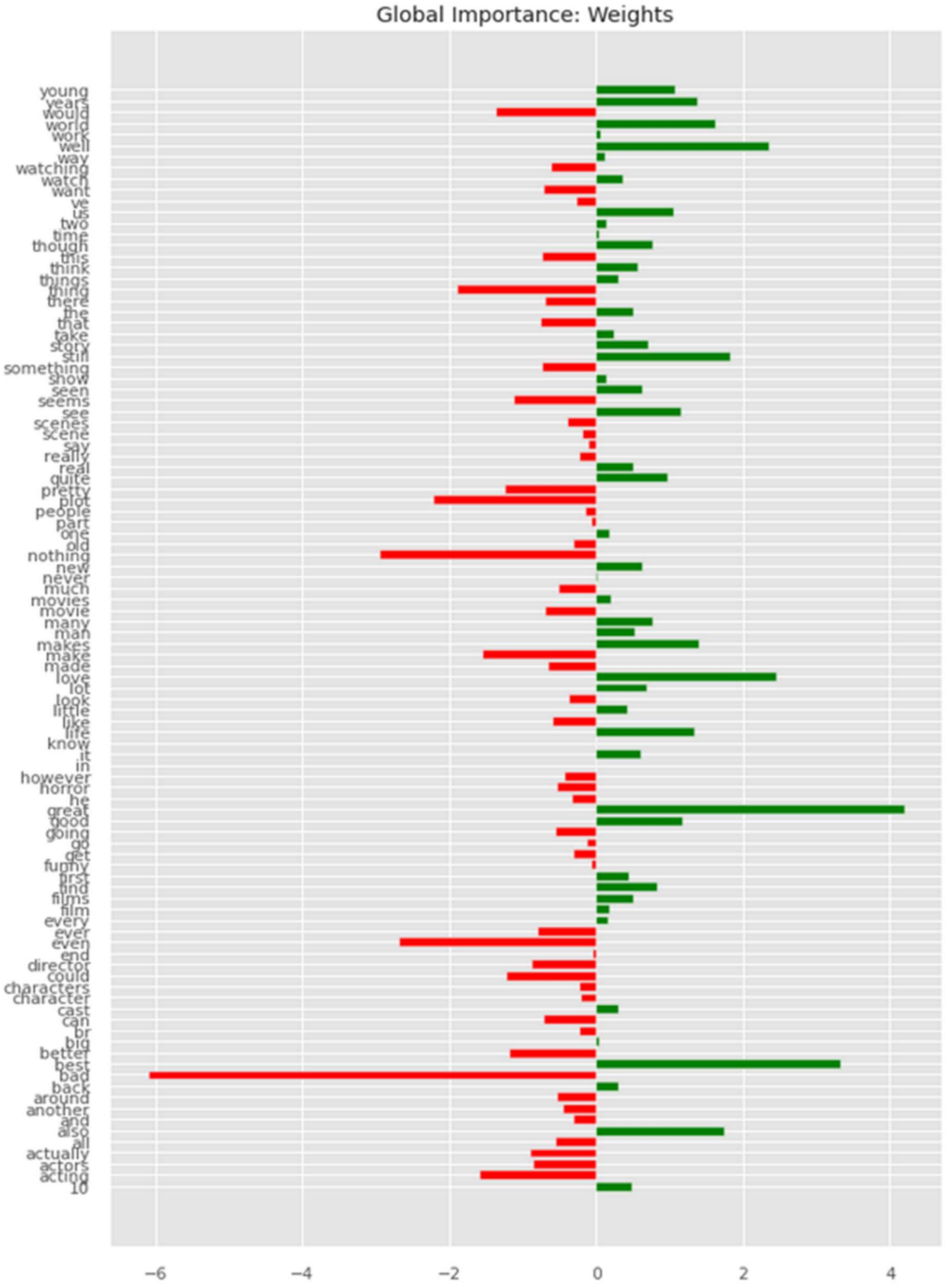
```
Model ROCAUC not calculated. Starting now
```

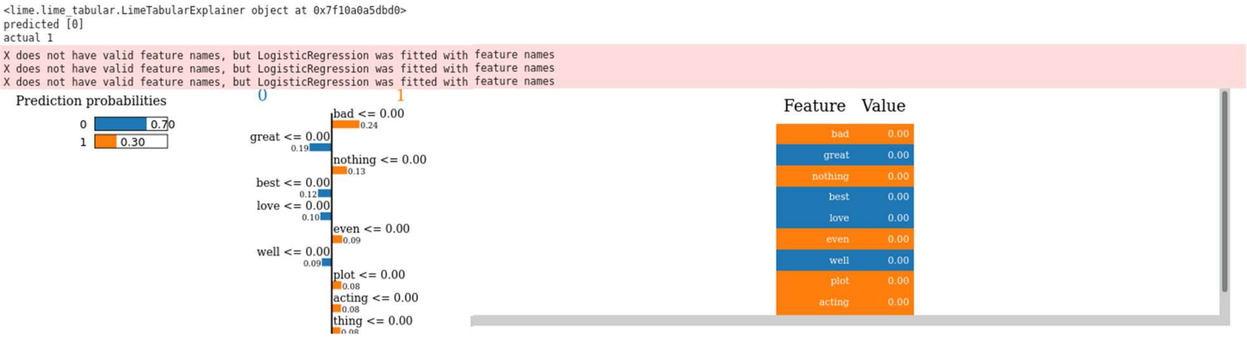


```
100% 101/101 [00:00<00:00, 4205.67it/s]
```




Feature Importances of Top 5 Features using XGBClassifier
bad
great
nothing
love
best
0
20
40
60
80
100
relative importance

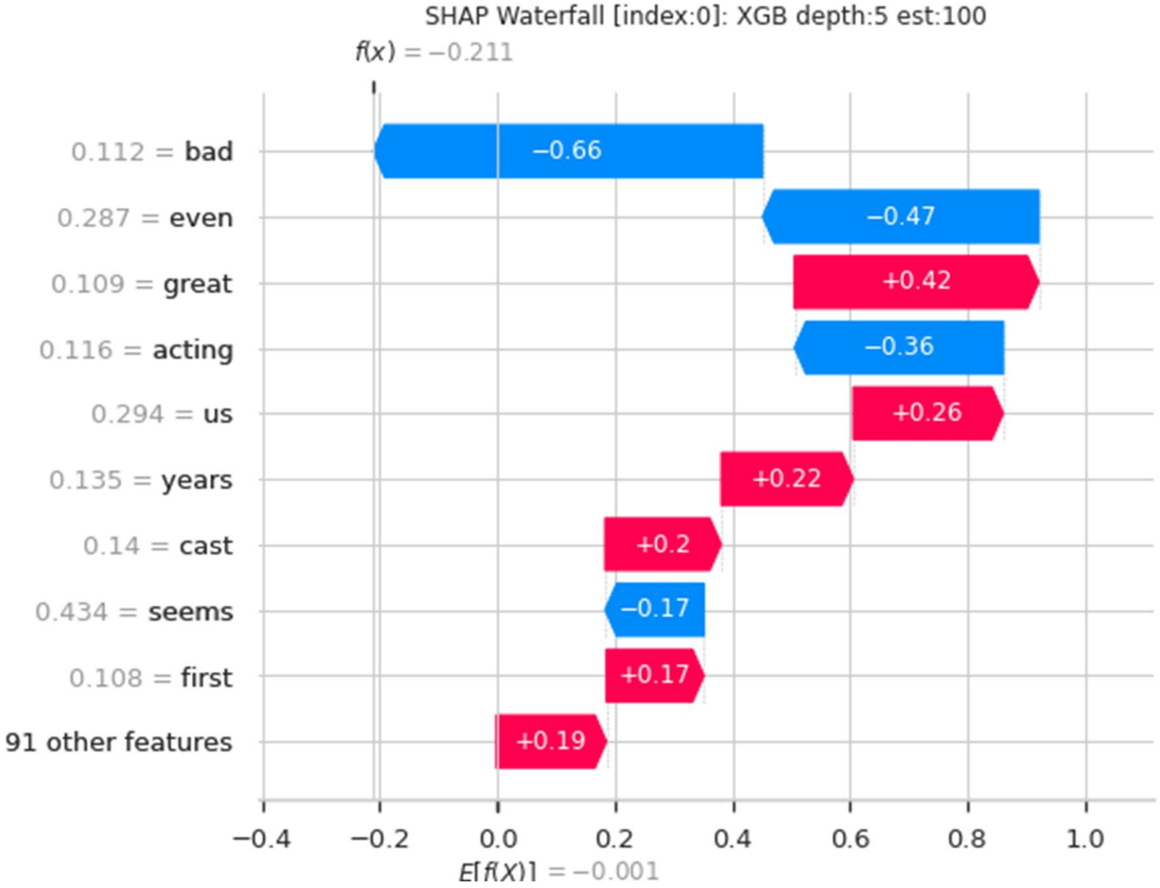

Global Importance: Weights
young
years
would
world
work
well
way
watching
watch
want
ve
us
two
time
though
this
think
things
thing
there
the
that
take
story
still
something
show
seen
seems
see
scenes
scene
say
really
real
quite
pretty
plot
people
part
one
old
nothing
new
never
much
movies
movie
many
man
makes
make
made
love
lot
look
little
like
life
know
it
in
however
horror
he
great
good
going
go
get
funny
first
find
films
film
every
ever
even
end
director
could
characters
character
cast
can
br
big
better
best
bad
back
around
another
and
also
all
actually
actors
acting
10
−6
−4
−2
0
2
4

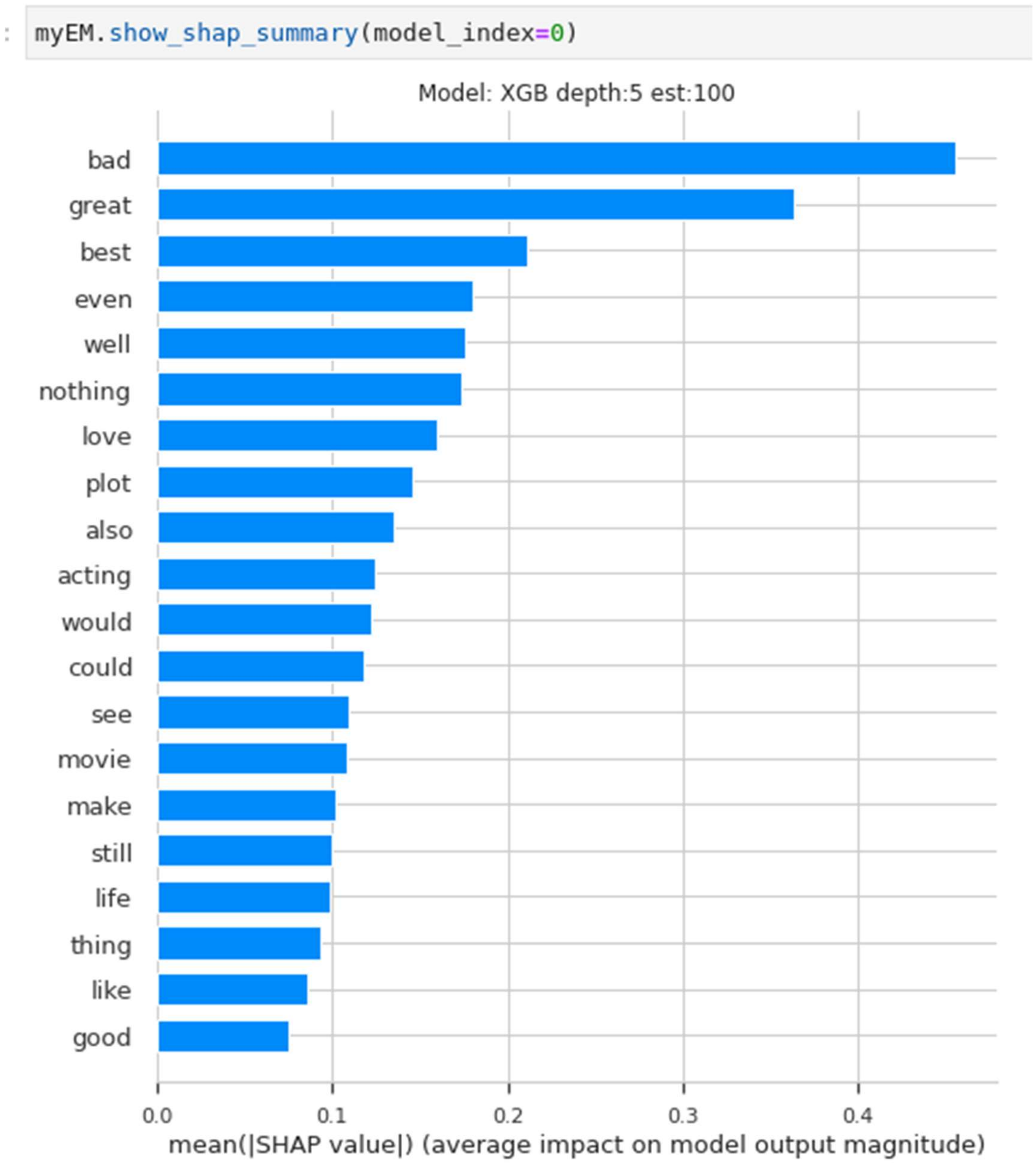


```
<lime.lime_tabular.LimeTabularExplainer object at 0x7f10a0a5dbd0>
predicted [0]
actual 1
X does not have valid feature names, but LogisticRegression was fitted with feature names
X does not have valid feature names, but LogisticRegression was fitted with feature names
X does not have valid feature names, but LogisticRegression was fitted with feature names
```

Feature	Value
bad	0.00
great	0.00
nothing	0.00
best	0.00
love	0.00
even	0.00
well	0.00
plot	0.00
acting	0.00

```
SHAP values not calculated. Generating now
Calculating shap_values for XGB depth:5 est:100
ntree_limit is deprecated, use `iteration_range` or model slicing instead.
```





```
: myEM.show_shap_summary(model_index=0)
```