INNOMATICS® RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Mlflow for experiment Tracking and Model Management

Made By:
Harsh Raj Gupta

About me

A world where chemical reactions whisper their secrets through data, where algorithms predict the perfect film for a cozy night in, and where nature's intricate patterns hold the key to optimizing processes. This is the world I see, the one ***I'm eager to build with the chisel of code and the mortar of machine learning*.**

I'm not just a Chemical Engineer in the making, I'm a data alchemist. I see molecules not just as building blocks, but as stories waiting to be told. Stories etched in numbers, patterns whispering with potential. And my tools? ***AI, ML, and data science – the incantations with which I translate these whispers into real-world solutions.***
My academic journey in Chemical Engineering has endowed me with a robust understanding of mathematical and engineering principles. Beyond my coursework, my passion for exploring the field of **data science** and **machine learning** has led me to delve into projects that bridge theory with tangible outcomes.

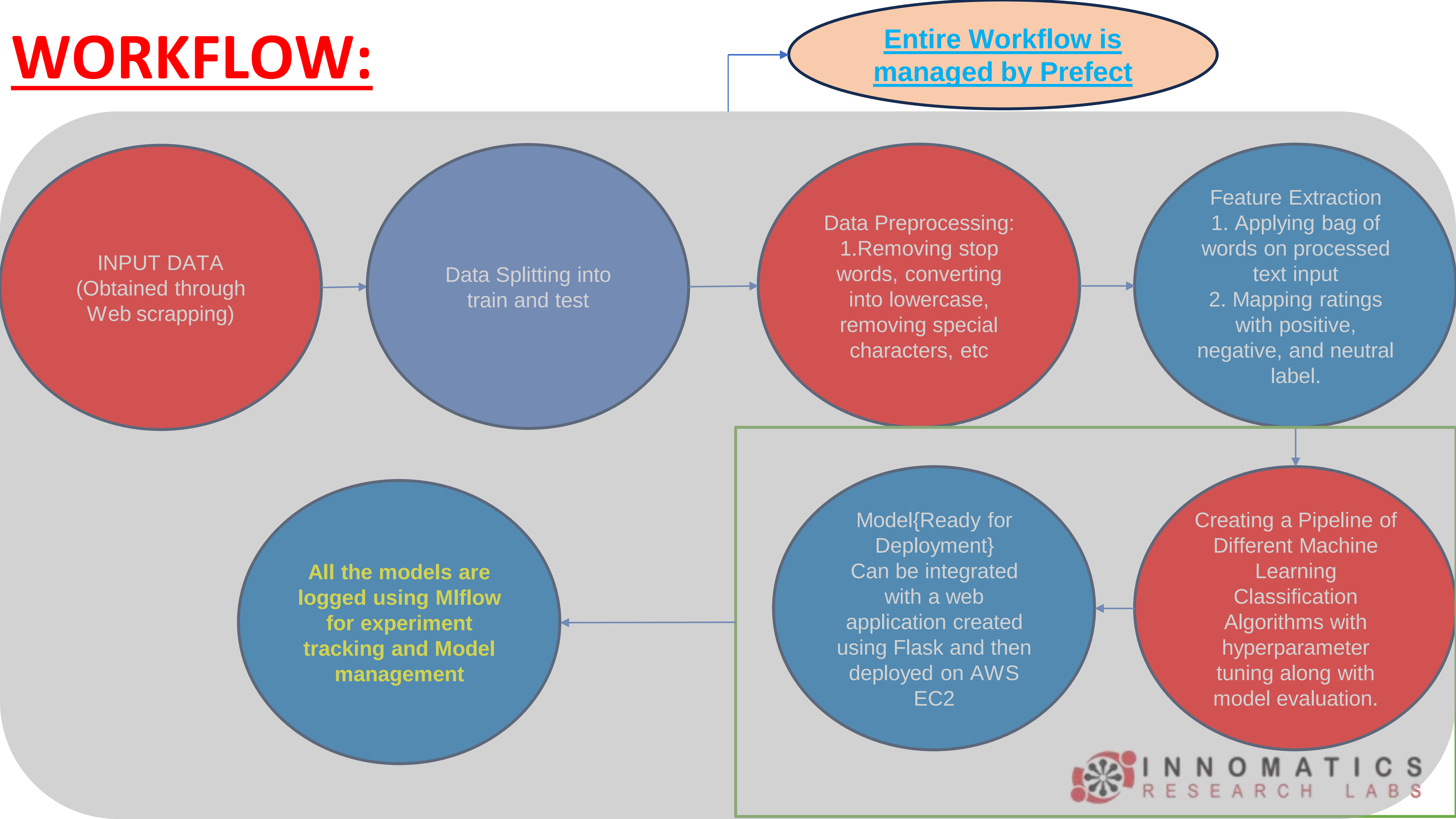
My GitHub repository showcases a diverse collection of collaborative ML-DL projects, ranging from **movie recommendations** to **face detection** and **generative adversarial networks**. Through these projects, I have honed my skills in translating theoretical concepts into practical, tangible outcomes

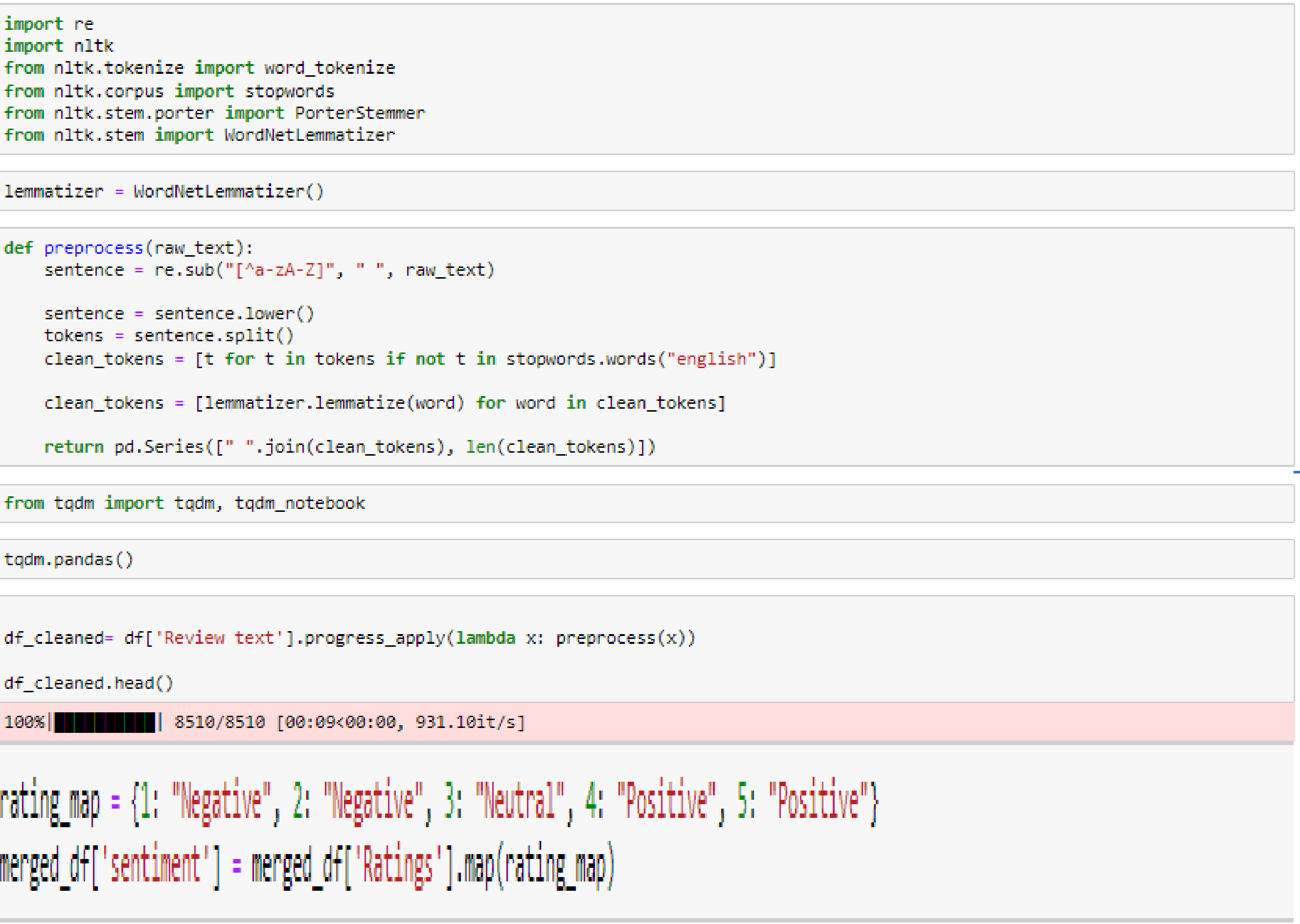
And it's not just about lines of code. Delving deeper into the code, I wield data structures like intricate tools, ***crafting stacks and queues that orchestrate calculations, sculpting efficient arrays and trees to organize information, and weaving algorithms like spells to uncover hidden patterns within data***. This mastery, honed in C++, translates seamlessly to Python's swift execution and Java's enterprise-grade robustness, empowering me to build solutions that are not just elegant, but also remarkably efficient.

A data alchemist ready to turn problems into possibilities, challenges into catalysts for change, fuelled by the power of code and data to unlock the universe's secrets.

INNOMATICS
RESEARCH LABS

WORKFLOW:


Entire Workflow is managed by Prefect
INPUT DATA (Obtained through Web scrapping)
Data Splitting into train and test
Data Preprocessing: 1.Removing stop words, converting into lowercase, removing special characters, etc
Feature Extraction 1. Applying bag of words on processed text input 2. Mapping ratings with positive, negative, and neutral label.
Creating a Pipeline of Different Machine Learning Classification Algorithms with hyperparameter tuning along with model evaluation.
Model{Ready for Deployment} Can be integrated with a web application created using Flask and then deployed on AWS EC2
All the models are logged using Mlflow for experiment tracking and Model management
INNOMATICS RESEARCH LABS



```
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
```

```
lemmatizer = WordNetLemmatizer()
```

```
def preprocess(raw_text):
    sentence = re.sub("[^a-zA-Z]", " ", raw_text)

    sentence = sentence.lower()
    tokens = sentence.split()
    clean_tokens = [t for t in tokens if not t in stopwords.words("english")]

    clean_tokens = [lemmatizer.lemmatize(word) for word in clean_tokens]

    return pd.Series([" ".join(clean_tokens), len(clean_tokens)])
```

```
from tqdm import tqdm, tqdm_notebook
```

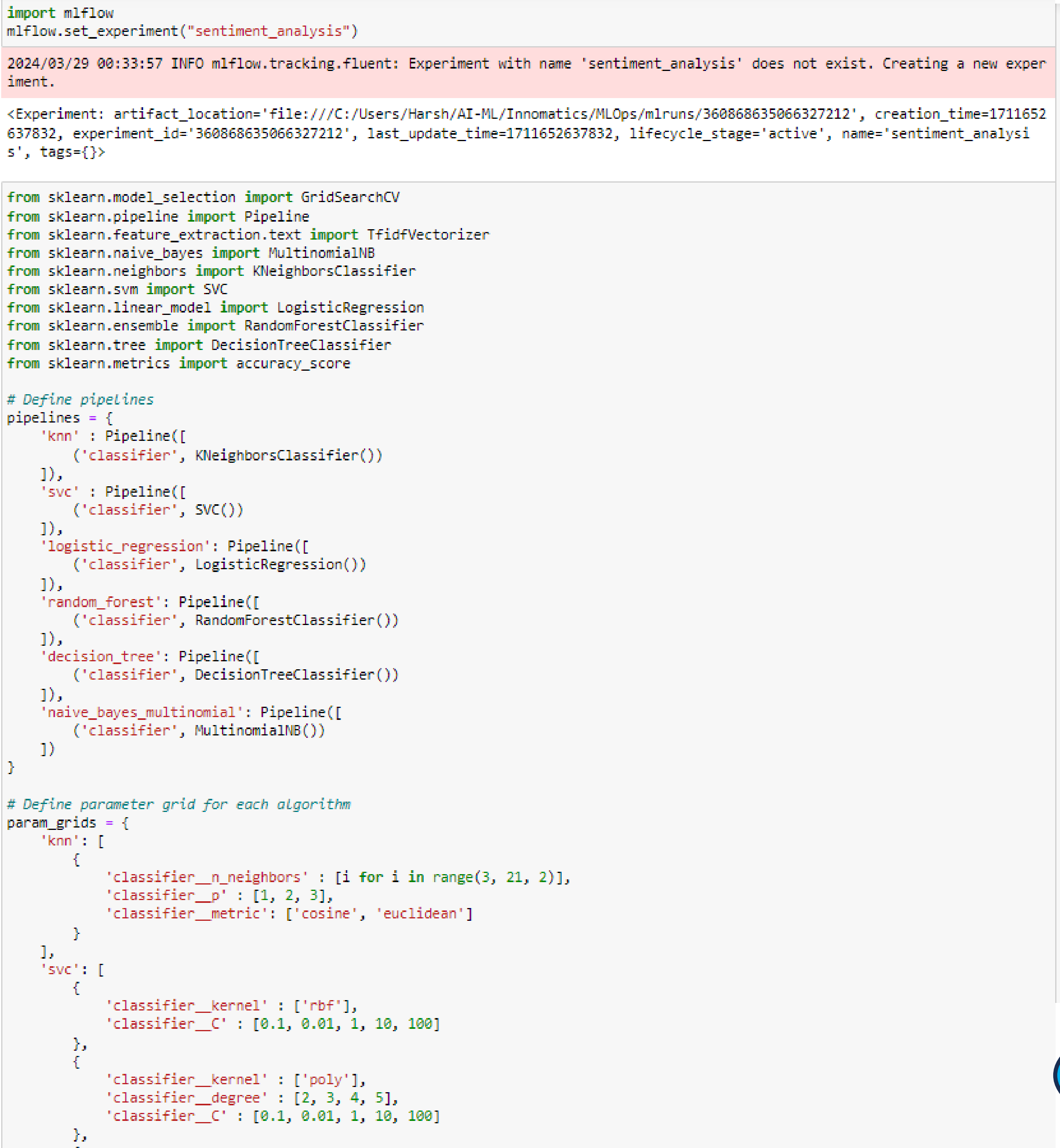
```
tqdm.pandas()
```

```
df_cleaned= df['Review text'].progress_apply(lambda x: preprocess(x))

df_cleaned.head()
```

```
100%|██████████| 8510/8510 [00:09<00:00, 931.10it/s]
```

```
rating_map = {1: "Negative", 2: "Negative", 3: "Neutral", 4: "Positive", 5: "Positive"}
merged_df['sentiment'] = merged_df['Ratings'].map(rating_map)
```

Data Preprocessing and Mapping of output with labels like positive, negative and neutral

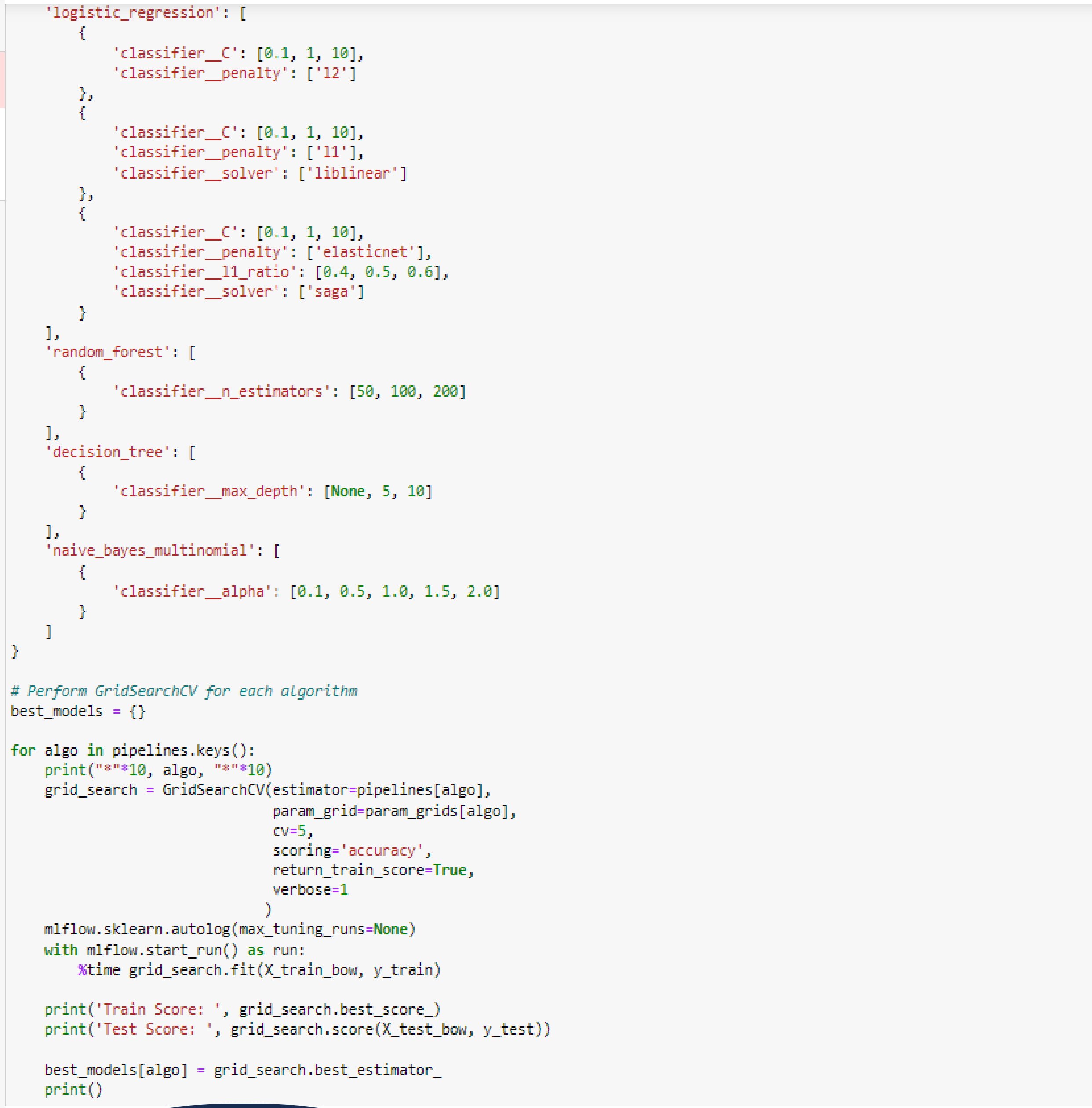


```
import mlflow
mlflow.set_experiment("sentiment_analysis")
```

```
2024/03/29 00:33:57 INFO mlflow.tracking.fluent: Experiment with name 'sentiment_analysis' does not exist. Creating a new exper
iment.
```

```
<Experiment: artifact_location='file:///C:/Users/Harsh/AI-ML/Innomatics/MLOps/mlruns/360868635066327212', creation_time=1711652
637832, experiment_id='360868635066327212', last_update_time=1711652637832, lifecycle_stage='active', name='sentiment_analysi
s', tags={}>
```

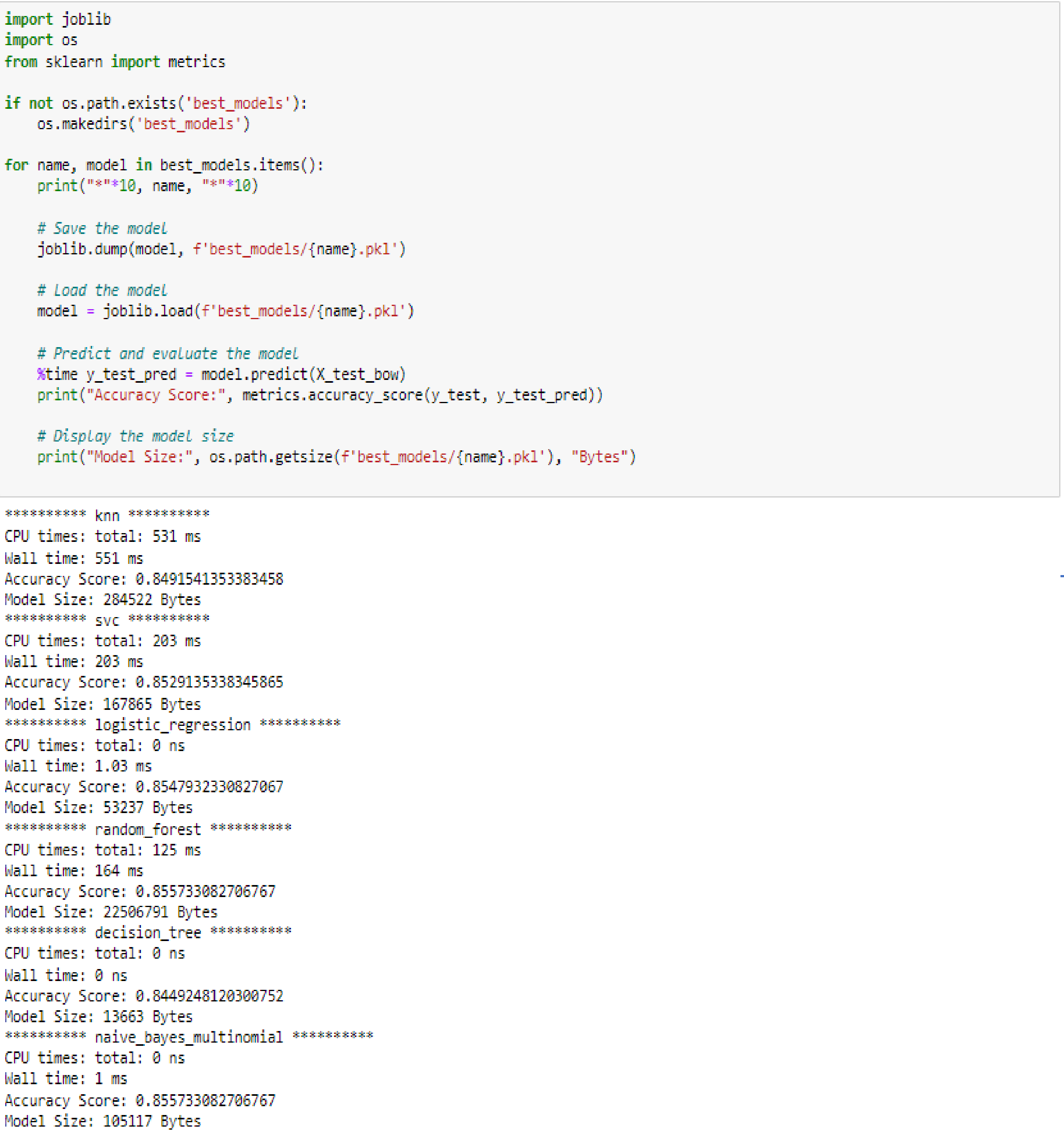
```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Define pipelines
pipelines = {
    'knn' : Pipeline([
        ('classifier', KNeighborsClassifier())
    ]),
    'svc' : Pipeline([
        ('classifier', SVC())
    ]),
    'logistic_regression': Pipeline([
        ('classifier', LogisticRegression())
    ]),
    'random_forest': Pipeline([
        ('classifier', RandomForestClassifier())
    ]),
    'decision_tree': Pipeline([
        ('classifier', DecisionTreeClassifier())
    ]),
    'naive_bayes_multinomial': Pipeline([
        ('classifier', MultinomialNB())
    ])
}

# Define parameter grid for each algorithm
param_grids = {
    'knn': [
        {
            'classifier__n_neighbors' : [i for i in range(3, 21, 2)],
            'classifier__p' : [1, 2, 3],
            'classifier__metric': ['cosine', 'euclidean']
        }
    ],
    'svc': [
        {
            'classifier__kernel' : ['rbf'],
            'classifier__C' : [0.1, 0.01, 1, 10, 100]
        },
        {
            'classifier__kernel' : ['poly'],
            'classifier__degree' : [2, 3, 4, 5],
            'classifier__C' : [0.1, 0.01, 1, 10, 100]
        },
```



```
    'logistic_regression': [
        {
            'classifier__C': [0.1, 1, 10],
            'classifier__penalty': ['l2']
        },
        {
            'classifier__C': [0.1, 1, 10],
            'classifier__penalty': ['l1'],
            'classifier__solver': ['liblinear']
        },
        {
            'classifier__C': [0.1, 1, 10],
            'classifier__penalty': ['elasticnet'],
            'classifier__l1_ratio': [0.4, 0.5, 0.6],
            'classifier__solver': ['saga']
        }
    ],
    'random_forest': [
        {
            'classifier__n_estimators': [50, 100, 200]
        }
    ],
    'decision_tree': [
        {
            'classifier__max_depth': [None, 5, 10]
        }
    ],
    'naive_bayes_multinomial': [
        {
            'classifier__alpha': [0.1, 0.5, 1.0, 1.5, 2.0]
        }
    ]
}

# Perform GridSearchCV for each algorithm
best_models = {}

for algo in pipelines.keys():
    print("*"*10, algo, "*"*10)
    grid_search = GridSearchCV(estimator=pipelines[algo],
                               param_grid=param_grids[algo],
                               cv=5,
                               scoring='accuracy',
                               return_train_score=True,
                               verbose=1
                              )
    mlflow.sklearn.autolog(max_tuning_runs=None)
    with mlflow.start_run() as run:
        %time grid_search.fit(X_train_bow, y_train)

    print('Train Score: ', grid_search.best_score_)
    print('Test Score: ', grid_search.score(X_test_bow, y_test))

    best_models[algo] = grid_search.best_estimator_
    print()
```

Model Pipeline

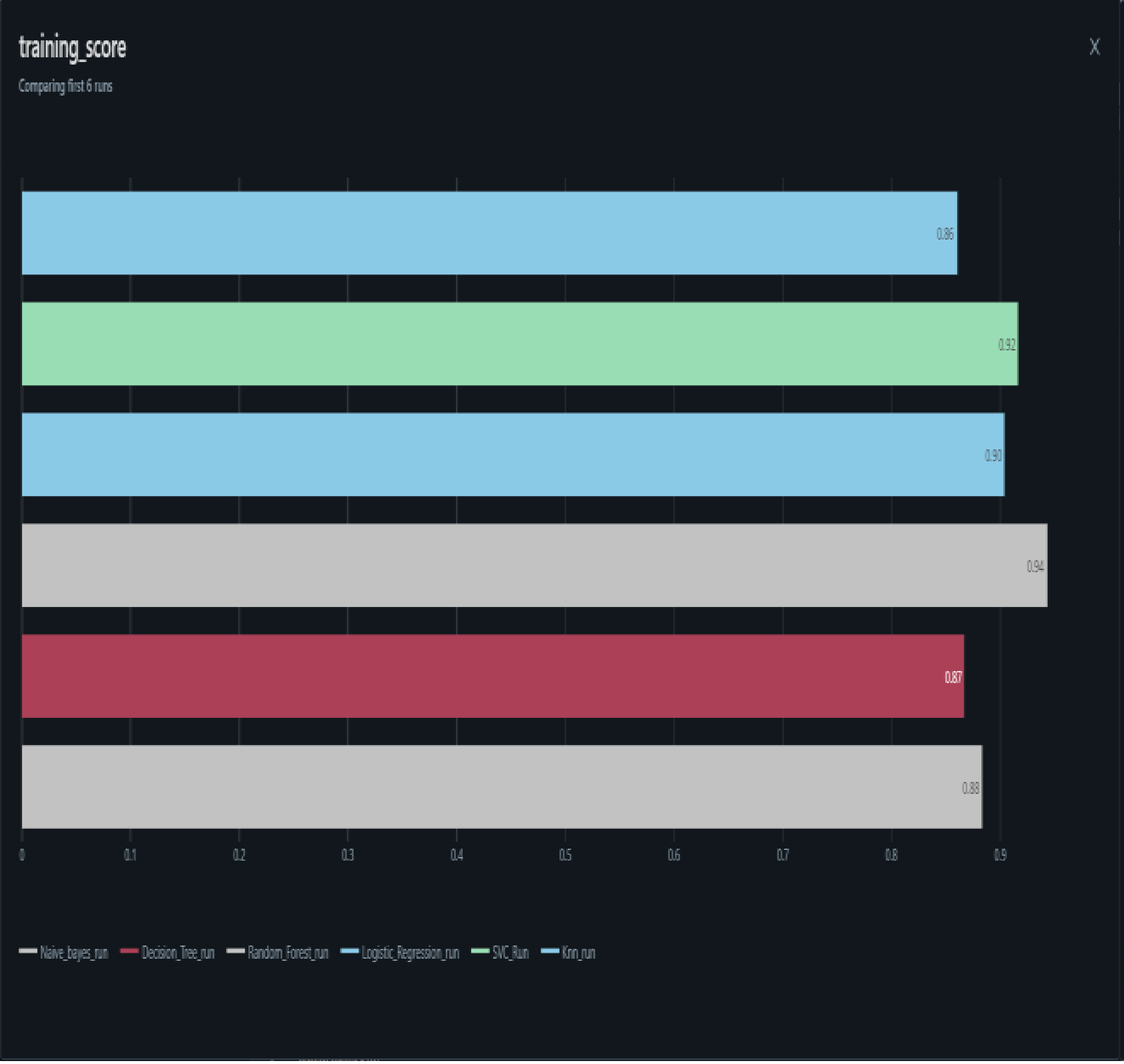
INNOMATICS
RESEARCH LABS

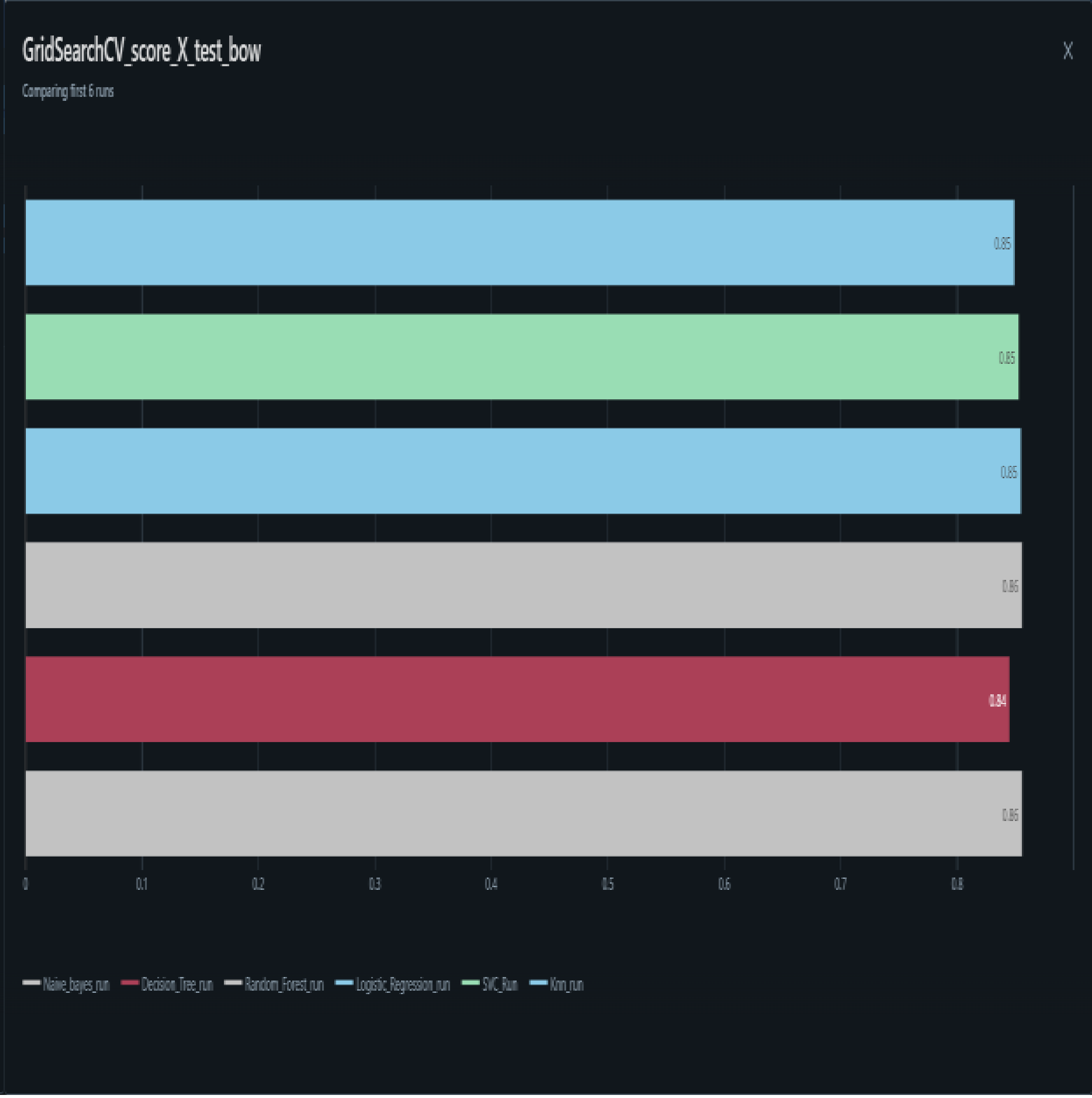


```python
import joblib
import os
from sklearn import metrics

if not os.path.exists('best_models'):
    os.makedirs('best_models')

for name, model in best_models.items():
    print("*"*10, name, "*"*10)

    # Save the model
    joblib.dump(model, f'best_models/{name}.pkl')

    # Load the model
    model = joblib.load(f'best_models/{name}.pkl')

    # Predict and evaluate the model
    %time y_test_pred = model.predict(X_test_bow)
    print("Accuracy Score:", metrics.accuracy_score(y_test, y_test_pred))

    # Display the model size
    print("Model Size:", os.path.getsize(f'best_models/{name}.pkl'), "Bytes")
```

```
********** knn **********
CPU times: total: 531 ms
Wall time: 551 ms
Accuracy Score: 0.8491541353383458
Model Size: 284522 Bytes
********** svc **********
CPU times: total: 203 ms
Wall time: 203 ms
Accuracy Score: 0.8529135338345865
Model Size: 167865 Bytes
********** logistic_regression **********
CPU times: total: 0 ns
Wall time: 1.03 ms
Accuracy Score: 0.8547932330827067
Model Size: 53237 Bytes
********** random_forest **********
CPU times: total: 125 ms
Wall time: 164 ms
Accuracy Score: 0.855733082706767
Model Size: 22506791 Bytes
********** decision_tree **********
CPU times: total: 0 ns
Wall time: 0 ns
Accuracy Score: 0.8449248120300752
Model Size: 13663 Bytes
********** naive_bayes_multinomial **********
CPU times: total: 0 ns
Wall time: 1 ms
Accuracy Score: 0.855733082706767
Model Size: 105117 Bytes
```

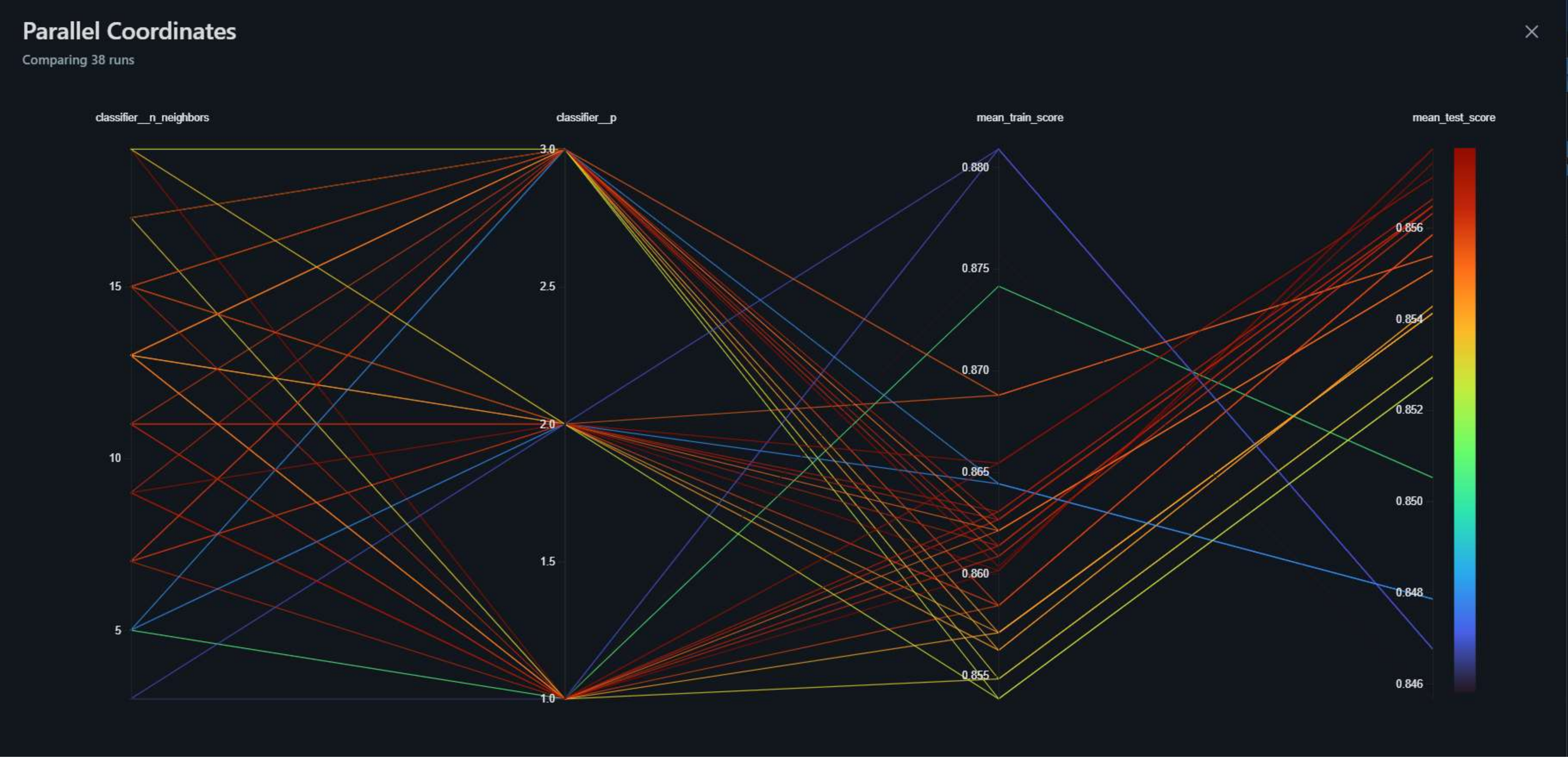
Result of All Models which have been trained on the data set along with their accuracy score



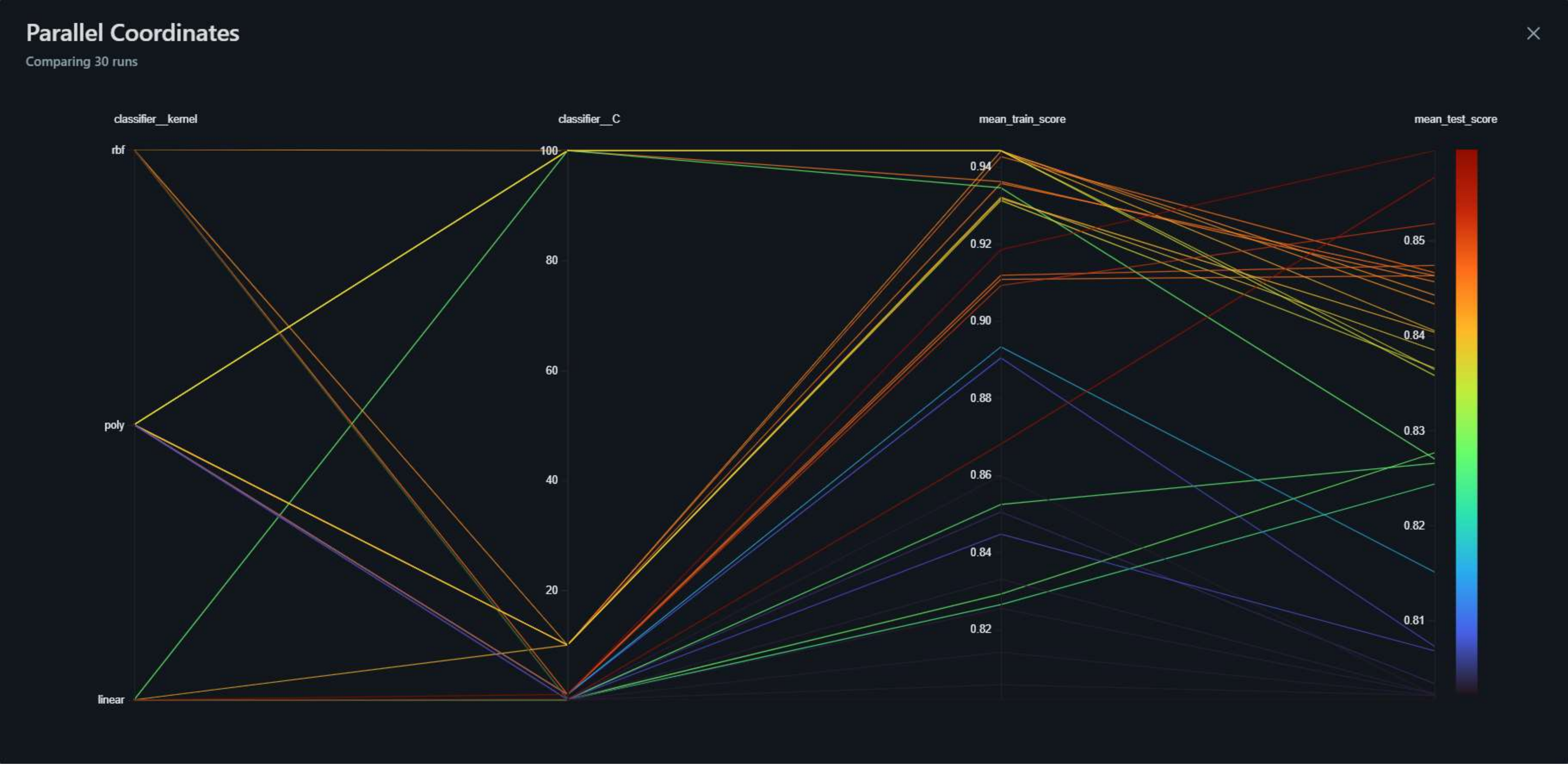


Training and Test Score of all the models

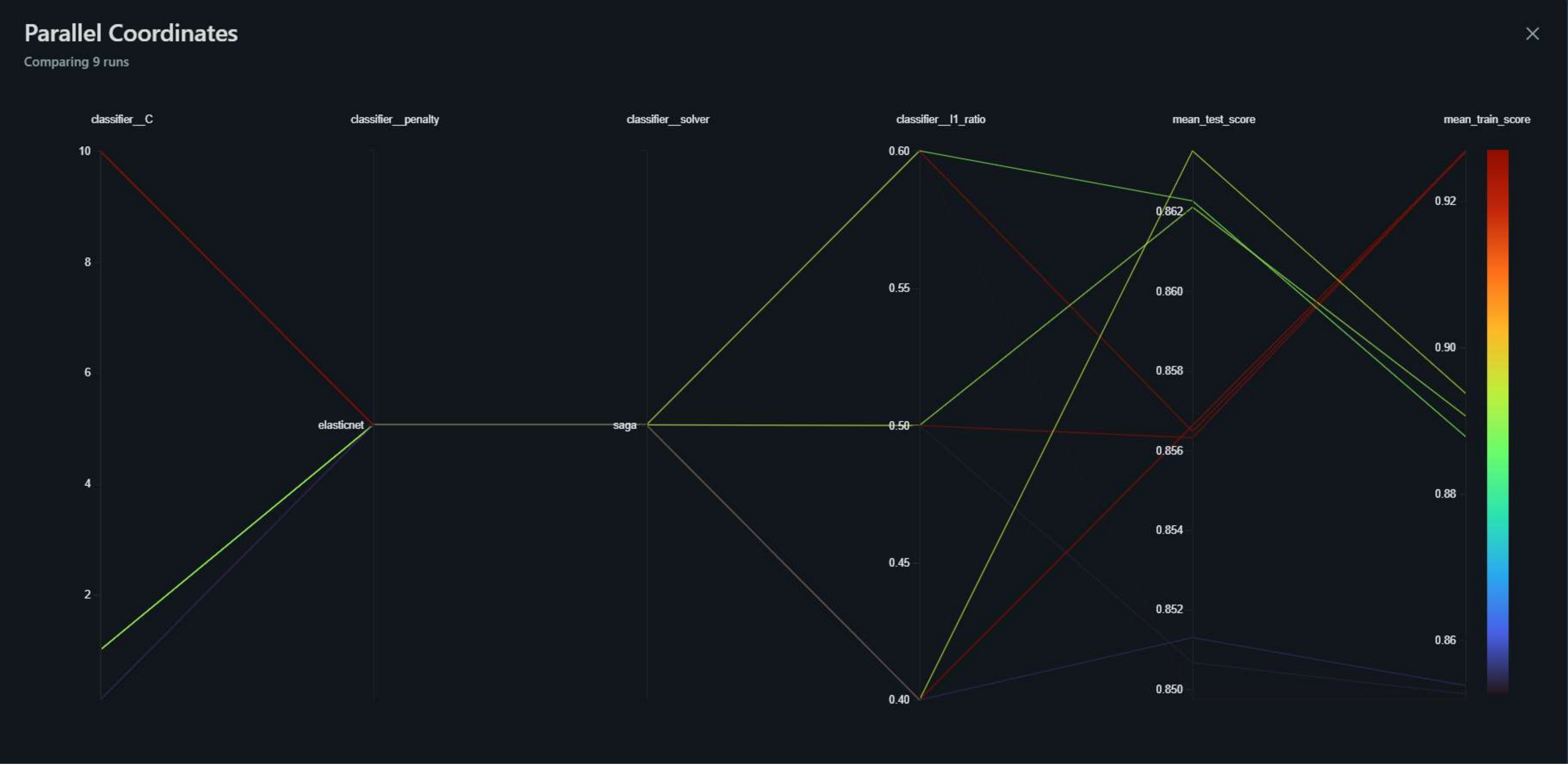
INNOMATICS RESEARCH LABS



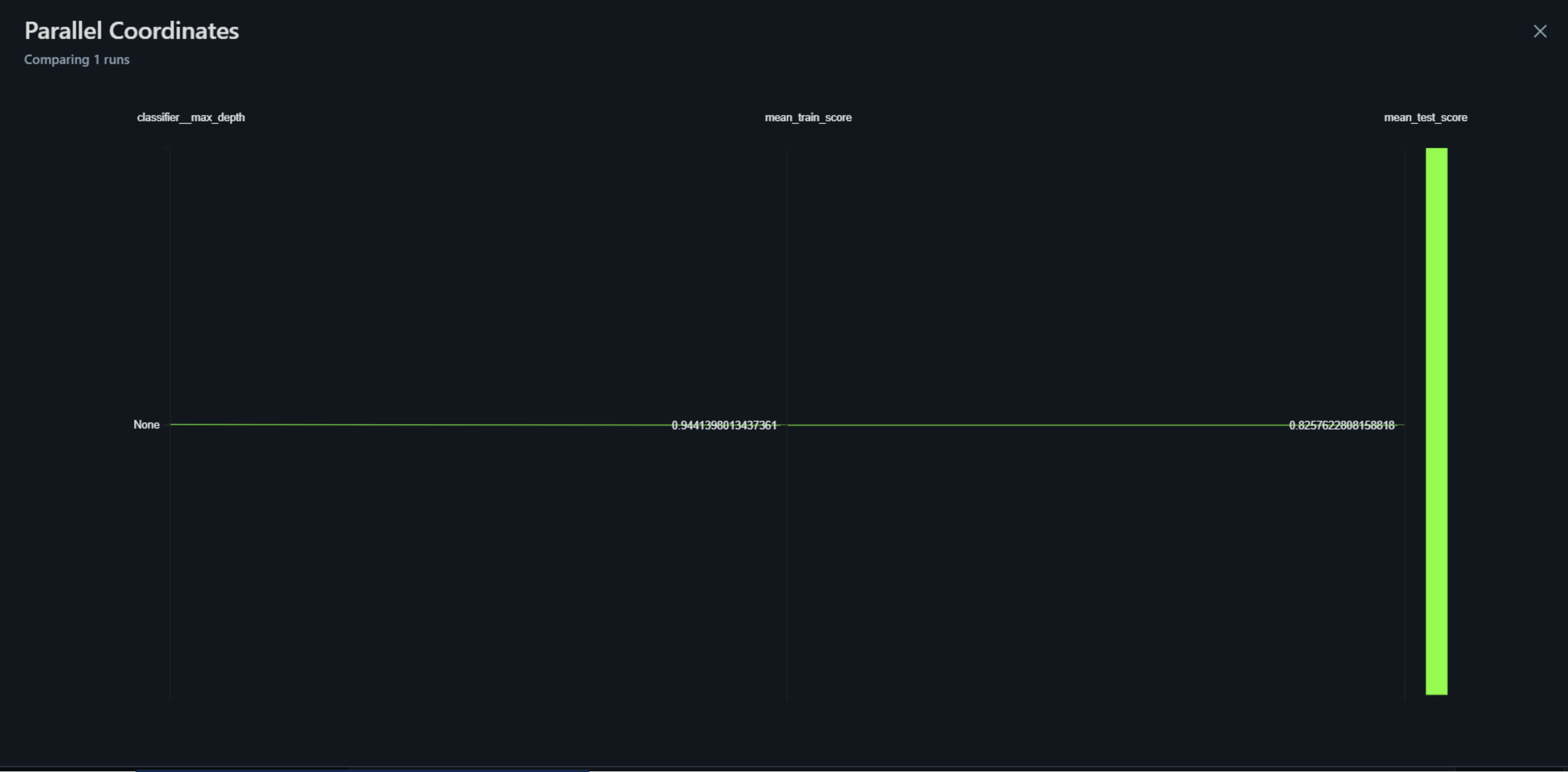
K_nearest Neighbor Classifier Model



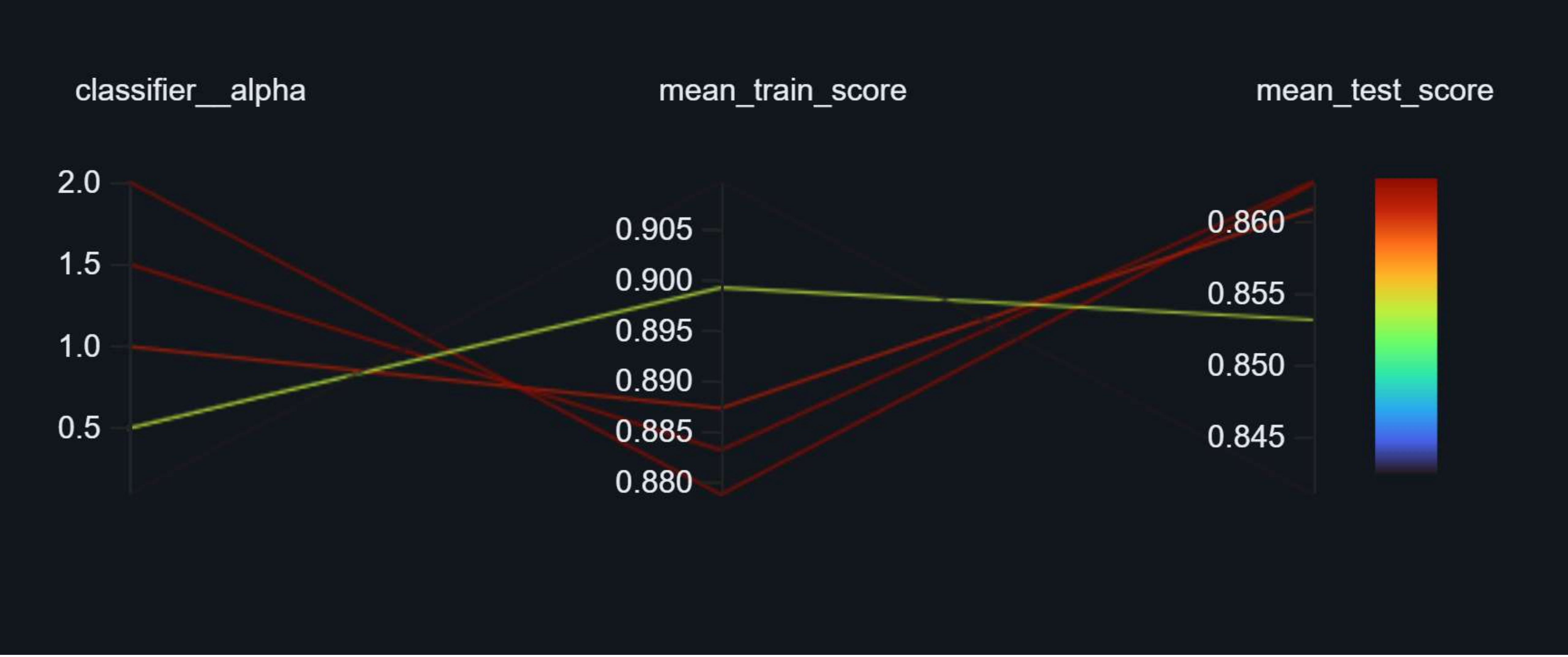
Support Vector Classifier Model



Logistic Regression Model

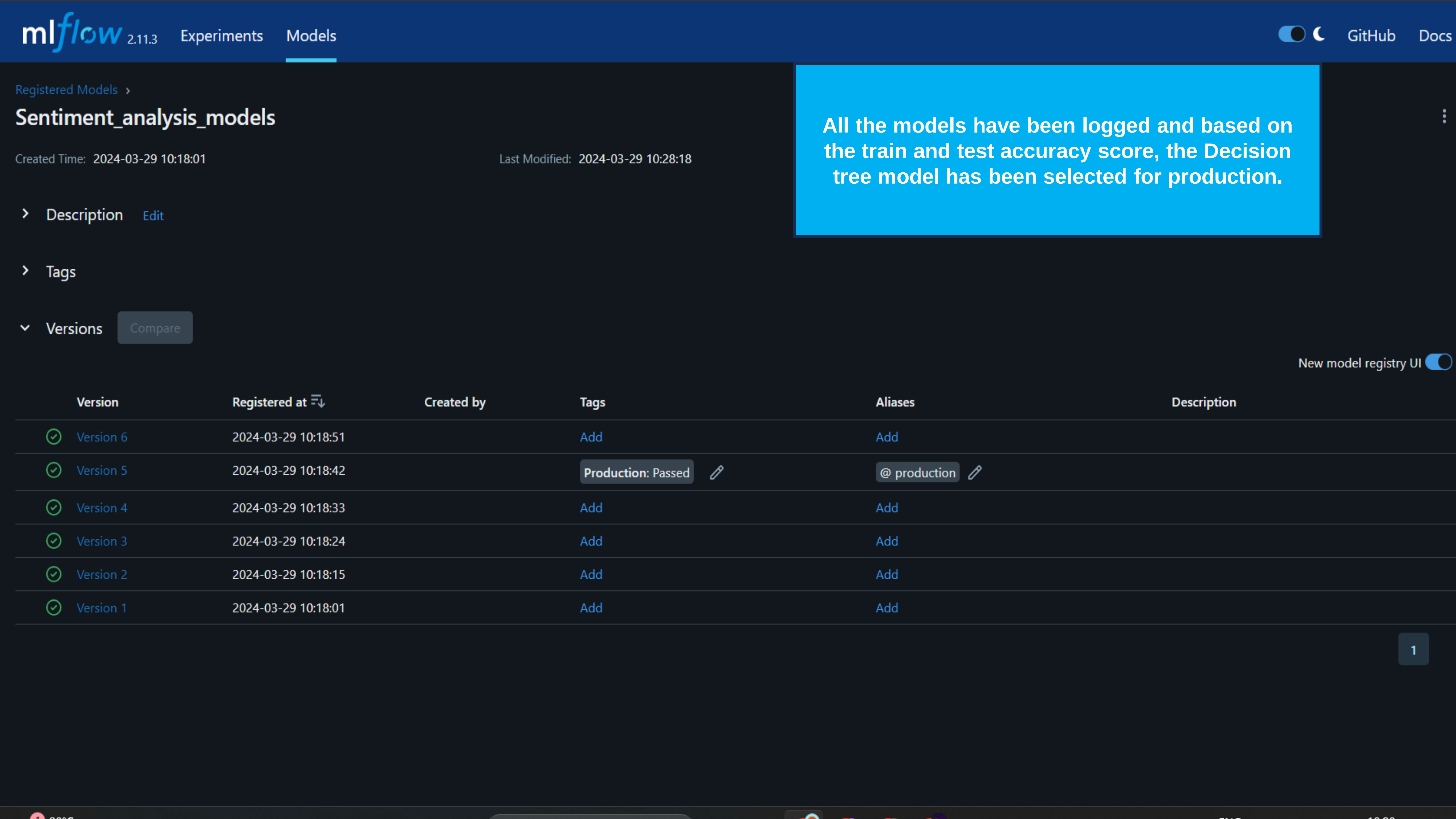


Decision Tree Model



Naive Bayes Model

INNOMATICS RESEARCH LABS



Registered Models >

Sentiment_analysis_models

Created Time: 2024-03-29 10:18:01   Last Modified: 2024-03-29 10:28:18

All the models have been logged and based on the train and test accuracy score, the Decision tree model has been selected for production.

> Description Edit

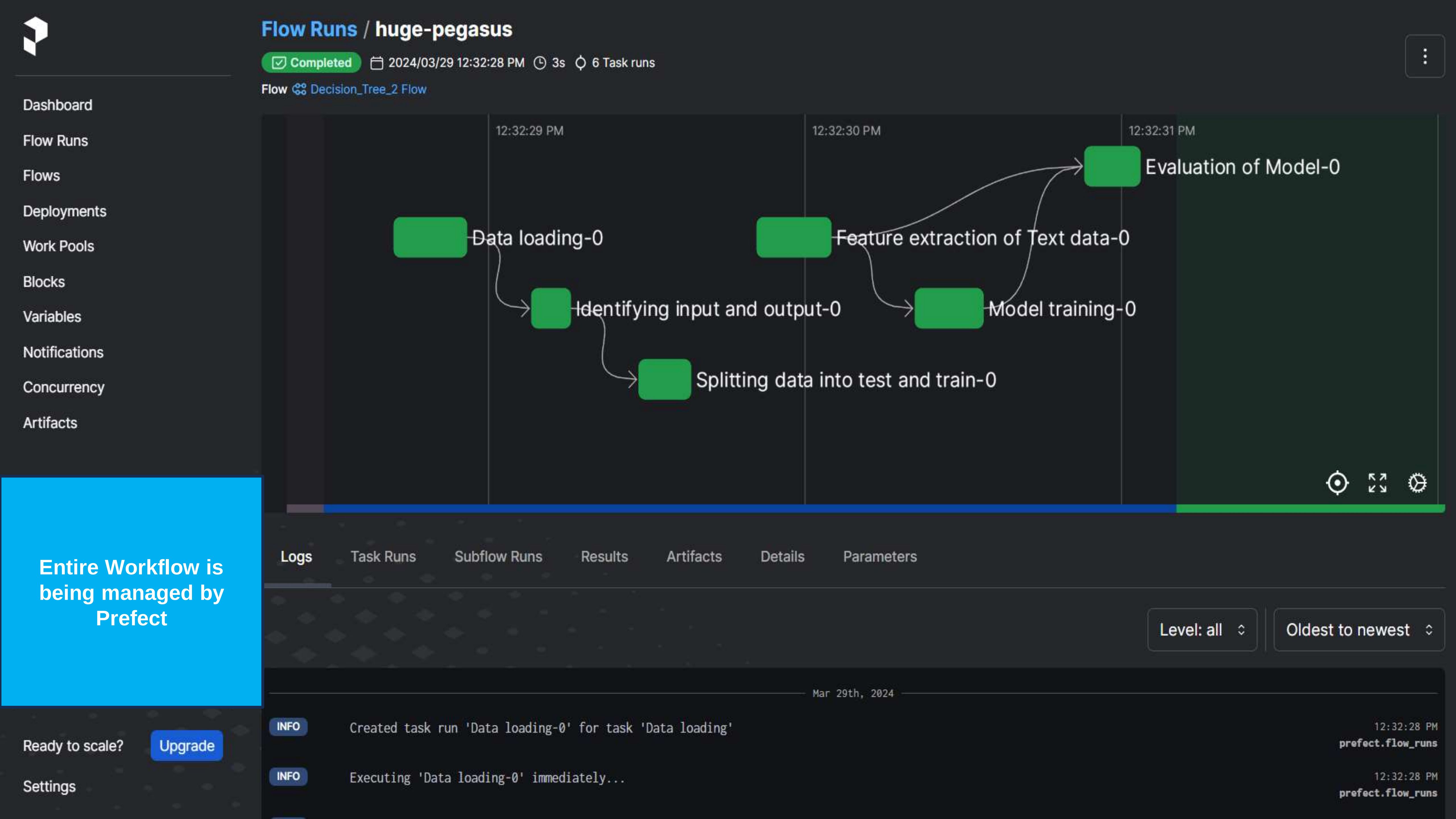
> Tags

Versions Compare

New model registry UI

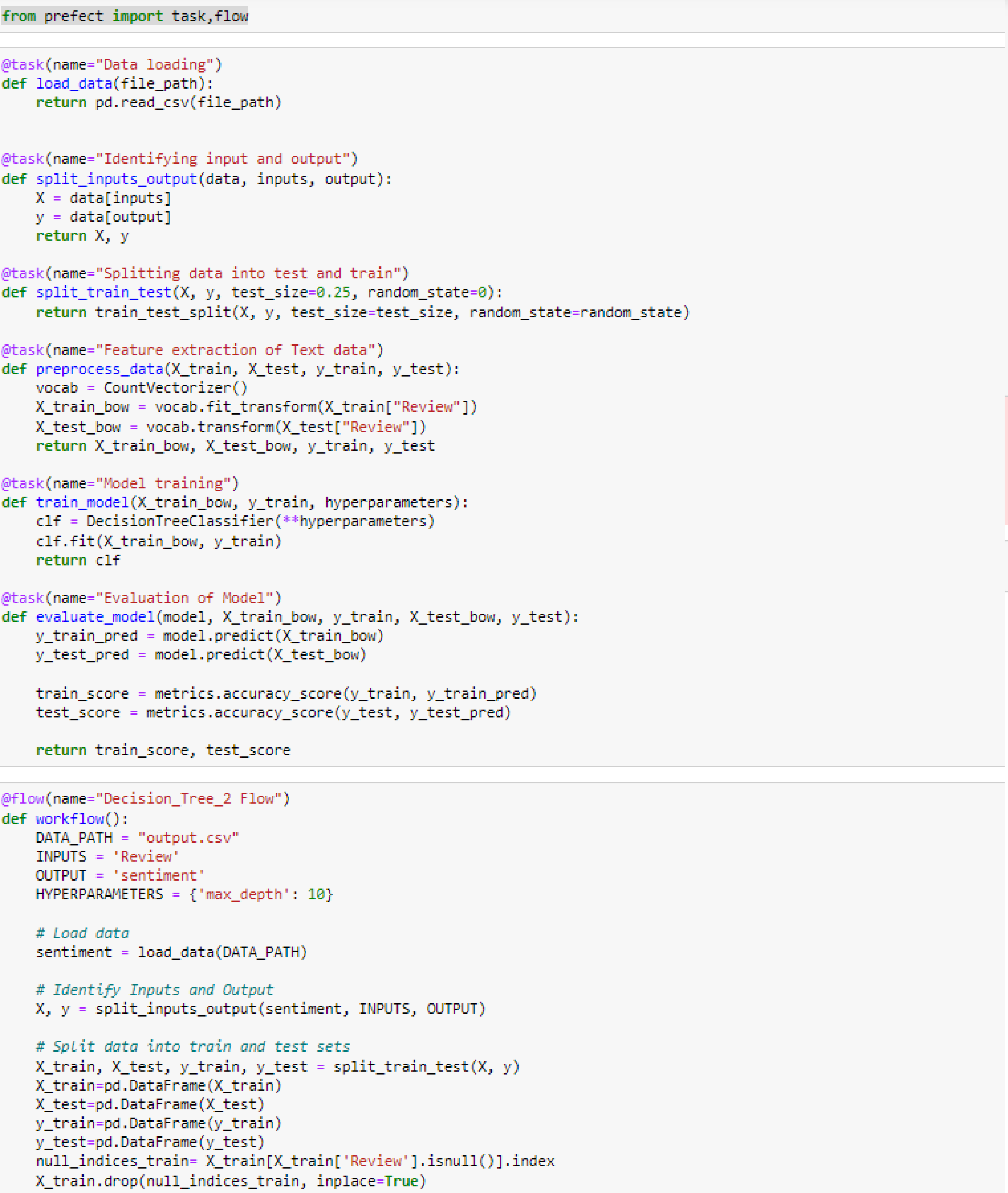
Version	Registered at	Created by	Tags	Aliases	Description
✓ Version 6	2024-03-29 10:18:51		Add	Add	
✓ Version 5	2024-03-29 10:18:42		Production: Passed	@ production	
✓ Version 4	2024-03-29 10:18:33		Add	Add	
✓ Version 3	2024-03-29 10:18:24		Add	Add	
✓ Version 2	2024-03-29 10:18:15		Add	Add	
✓ Version 1	2024-03-29 10:18:01		Add	Add	

1



```
Mar 29th, 2024
INFO  Created task run 'Data loading-0' for task 'Data loading'    12:32:28 PM
                                                                    prefect.flow_runs
INFO  Executing 'Data loading-0' immediately...                    12:32:28 PM
                                                                    prefect.flow_runs
```

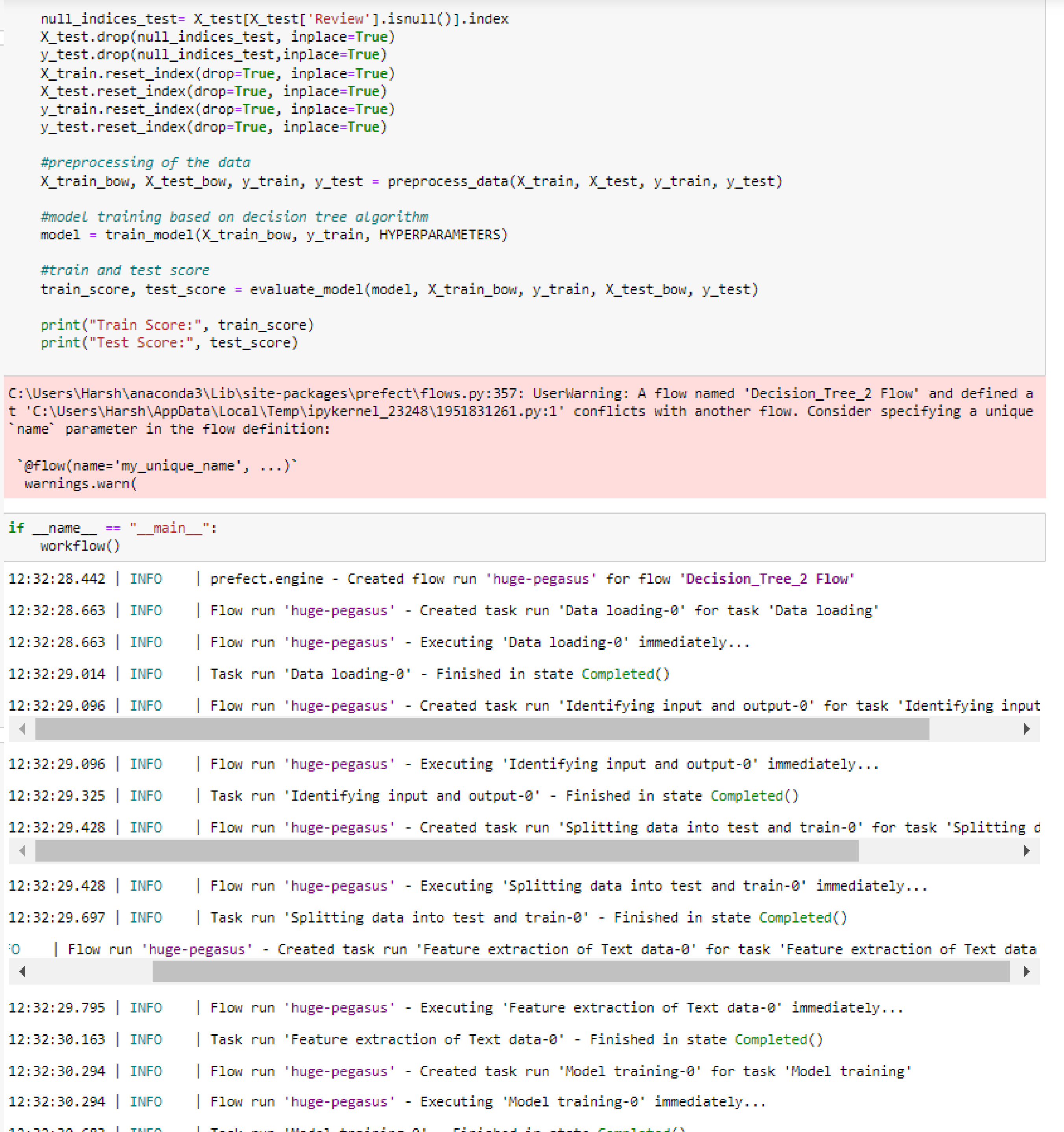
Entire Workflow is being managed by Prefect



```python
from prefect import task,flow
```

```python
@task(name="Data loading")
def load_data(file_path):
    return pd.read_csv(file_path)


@task(name="Identifying input and output")
def split_inputs_output(data, inputs, output):
    X = data[inputs]
    y = data[output]
    return X, y

@task(name="Splitting data into test and train")
def split_train_test(X, y, test_size=0.25, random_state=0):
    return train_test_split(X, y, test_size=test_size, random_state=random_state)

@task(name="Feature extraction of Text data")
def preprocess_data(X_train, X_test, y_train, y_test):
    vocab = CountVectorizer()
    X_train_bow = vocab.fit_transform(X_train["Review"])
    X_test_bow = vocab.transform(X_test["Review"])
    return X_train_bow, X_test_bow, y_train, y_test

@task(name="Model training")
def train_model(X_train_bow, y_train, hyperparameters):
    clf = DecisionTreeClassifier(**hyperparameters)
    clf.fit(X_train_bow, y_train)
    return clf

@task(name="Evaluation of Model")
def evaluate_model(model, X_train_bow, y_train, X_test_bow, y_test):
    y_train_pred = model.predict(X_train_bow)
    y_test_pred = model.predict(X_test_bow)

    train_score = metrics.accuracy_score(y_train, y_train_pred)
    test_score = metrics.accuracy_score(y_test, y_test_pred)

    return train_score, test_score
```

```python
@flow(name="Decision_Tree_2 Flow")
def workflow():
    DATA_PATH = "output.csv"
    INPUTS = 'Review'
    OUTPUT = 'sentiment'
    HYPERPARAMETERS = {'max_depth': 10}

    # Load data
    sentiment = load_data(DATA_PATH)

    # Identify Inputs and Output
    X, y = split_inputs_output(sentiment, INPUTS, OUTPUT)

    # Split data into train and test sets
    X_train, X_test, y_train, y_test = split_train_test(X, y)
    X_train=pd.DataFrame(X_train)
    X_test=pd.DataFrame(X_test)
    y_train=pd.DataFrame(y_train)
    y_test=pd.DataFrame(y_test)
    null_indices_train= X_train[X_train['Review'].isnull()].index
    X_train.drop(null_indices_train, inplace=True)
```



```python
    null_indices_test= X_test[X_test['Review'].isnull()].index
    X_test.drop(null_indices_test, inplace=True)
    y_test.drop(null_indices_test,inplace=True)
    X_train.reset_index(drop=True, inplace=True)
    X_test.reset_index(drop=True, inplace=True)
    y_train.reset_index(drop=True, inplace=True)
    y_test.reset_index(drop=True, inplace=True)

    #preprocessing of the data
    X_train_bow, X_test_bow, y_train, y_test = preprocess_data(X_train, X_test, y_train, y_test)

    #model training based on decision tree algorithm
    model = train_model(X_train_bow, y_train, HYPERPARAMETERS)

    #train and test score
    train_score, test_score = evaluate_model(model, X_train_bow, y_train, X_test_bow, y_test)

    print("Train Score:", train_score)
    print("Test Score:", test_score)
```

```
C:\Users\Harsh\anaconda3\Lib\site-packages\prefect\flows.py:357: UserWarning: A flow named 'Decision_Tree_2 Flow' and defined a
t 'C:\Users\Harsh\AppData\Local\Temp\ipykernel_23248\1951831261.py:1' conflicts with another flow. Consider specifying a unique
`name` parameter in the flow definition:

 `@flow(name='my_unique_name', ...)`
  warnings.warn(
```

```python
if __name__ == "__main__":
    workflow()
```

```
12:32:28.442 | INFO    | prefect.engine - Created flow run 'huge-pegasus' for flow 'Decision_Tree_2 Flow'
12:32:28.663 | INFO    | Flow run 'huge-pegasus' - Created task run 'Data loading-0' for task 'Data loading'
12:32:28.663 | INFO    | Flow run 'huge-pegasus' - Executing 'Data loading-0' immediately...
12:32:29.014 | INFO    | Task run 'Data loading-0' - Finished in state Completed()
12:32:29.096 | INFO    | Flow run 'huge-pegasus' - Created task run 'Identifying input and output-0' for task 'Identifying input
12:32:29.096 | INFO    | Flow run 'huge-pegasus' - Executing 'Identifying input and output-0' immediately...
12:32:29.325 | INFO    | Task run 'Identifying input and output-0' - Finished in state Completed()
12:32:29.428 | INFO    | Flow run 'huge-pegasus' - Created task run 'Splitting data into test and train-0' for task 'Splitting d
12:32:29.428 | INFO    | Flow run 'huge-pegasus' - Executing 'Splitting data into test and train-0' immediately...
12:32:29.697 | INFO    | Task run 'Splitting data into test and train-0' - Finished in state Completed()
O    | Flow run 'huge-pegasus' - Created task run 'Feature extraction of Text data-0' for task 'Feature extraction of Text data
12:32:29.795 | INFO    | Flow run 'huge-pegasus' - Executing 'Feature extraction of Text data-0' immediately...
12:32:30.163 | INFO    | Task run 'Feature extraction of Text data-0' - Finished in state Completed()
12:32:30.294 | INFO    | Flow run 'huge-pegasus' - Created task run 'Model training-0' for task 'Model training'
12:32:30.294 | INFO    | Flow run 'huge-pegasus' - Executing 'Model training-0' immediately...
12:32:30.683 | INFO    | Task run 'Model training-0' - Finished in state Completed()
```

THANK YOU

INNOMATICS
RESEARCH LABS