

## MLFlow Project Report on Titanic Dataset

### Steps:

1. Install MLFlow and required libraries.

Run the following commands in terminal:

```
pip install mlflow  
pip install scikit-learn
```

2. Load, preprocess the data and perform EDA.

### **EDA Insights :**

- Only **38%** of the passengers **survived**
- Did any **gender** have a better chance of survival?
  - **Females** had a survival rate of **74%**
  - **Males** had a survival rate of **19%**
- Did belonging a particular **socio-economic class** have an impact chances of survival?
  - **Upper class** had a survival rate of **63%**
  - **Middle class** had a survival rate of **47%**
  - **Lower class** had a survival rate of **41%**
- Did people with **family** have a better chance of surviving?
  - People **with Siblings/spouses** had roughly **50%** probability of surviving
  - People **without Siblings/spouses** had a **35%** probability of surviving
  - People with Siblings/spouses had a better chance of surviving probably because they might have been given preference.
  - Similar trend can be seen for people with parents/children v/s those without.

Preprocessing and EDA have been documented in the following file:

Titantic\_preprocessing.ipynb

(Link : [https://github.com/h-mehta/SE-for-DS/blob/main/Data%20Engineering/MLFlow/titanic/Titantic\\_preprocessing.ipynb](https://github.com/h-mehta/SE-for-DS/blob/main/Data%20Engineering/MLFlow/titanic/Titantic_preprocessing.ipynb) )

3. Save the preprocessed data so it can be directly read-in by the .py file that contains code to develop model on this pre-processed data.

Data after pre-processing: titanic\_processed\_data.csv

(Link: [https://github.com/h-mehta/SE-for-DS/blob/main/Data%20Engineering/MLFlow/titanic/titanic\\_processed\\_data.csv](https://github.com/h-mehta/SE-for-DS/blob/main/Data%20Engineering/MLFlow/titanic/titanic_processed_data.csv) )

4. Train the classification model and log the parameters and metrics after each run using MLFlow.

Code in file: titanic\_code.py

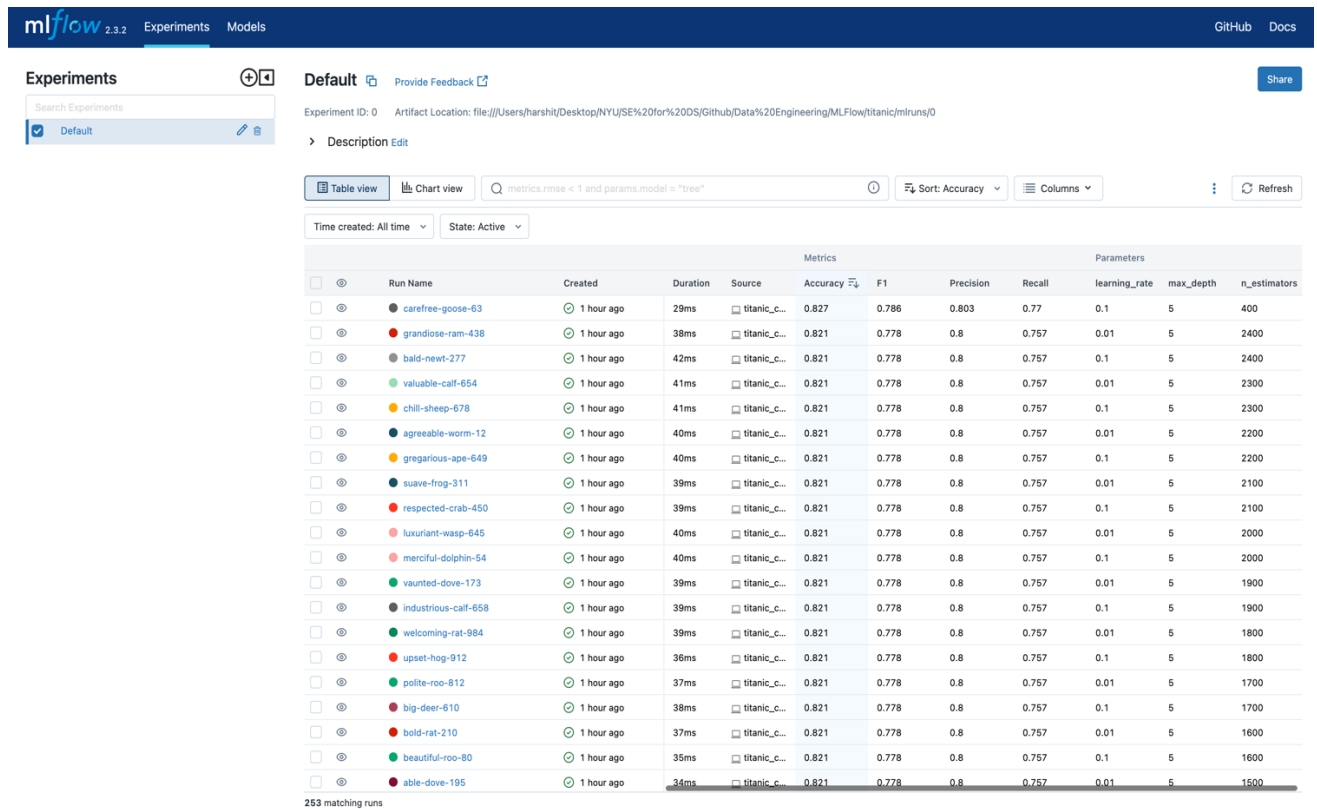
(Link: [https://github.com/h-mehta/SE-for-DS/blob/main/Data%20Engineering/MLFlow/titanic/titanic\\_code.py](https://github.com/h-mehta/SE-for-DS/blob/main/Data%20Engineering/MLFlow/titanic/titanic_code.py) )

5. Can create custom logs also apart from those generated by MLFlow:

```
2023-06-05 16:57:58,158 - __main__ - INFO - -----
2023-06-05 16:57:59,938 - __main__ - WARNING - Start of new run ...
2023-06-05 16:58:01,439 - __main__ - INFO - -----
2023-06-05 16:58:01,439 - __main__ - INFO - PARAMETERS : ---
2023-06-05 16:58:01,439 - __main__ - INFO - n_estimators : 2400
2023-06-05 16:58:01,439 - __main__ - INFO - learning_rate : 0.05
2023-06-05 16:58:01,439 - __main__ - INFO - max_depth : 7
2023-06-05 16:58:01,439 - __main__ - INFO - METRICS : ---
2023-06-05 16:58:01,439 - __main__ - INFO - Accuracy : 0.7877094972067039
2023-06-05 16:58:01,439 - __main__ - INFO - Precision : 0.7647058823529411
2023-06-05 16:58:01,439 - __main__ - INFO - Recall : 0.7027027027027027
2023-06-05 16:58:01,439 - __main__ - INFO - F1 : 0.7323943661971832
2023-06-05 16:58:01,440 - __main__ - INFO - -----
2023-06-05 16:58:03,216 - __main__ - WARNING - Start of new run ...
2023-06-05 16:58:04,888 - __main__ - INFO - -----
2023-06-05 16:58:04,889 - __main__ - INFO - PARAMETERS : ---
2023-06-05 16:58:04,889 - __main__ - INFO - n_estimators : 2400
2023-06-05 16:58:04,889 - __main__ - INFO - learning_rate : 0.01
2023-06-05 16:58:04,889 - __main__ - INFO - max_depth : 5
2023-06-05 16:58:04,889 - __main__ - INFO - METRICS : ---
2023-06-05 16:58:04,889 - __main__ - INFO - Accuracy : 0.8212290502793296
2023-06-05 16:58:04,889 - __main__ - INFO - Precision : 0.8
2023-06-05 16:58:04,889 - __main__ - INFO - Recall : 0.7567567567567568
2023-06-05 16:58:04,889 - __main__ - INFO - F1 : 0.7777777777777778
2023-06-05 16:58:04,889 - __main__ - INFO - -----
2023-06-05 16:58:06,670 - __main__ - WARNING - Start of new run ...
2023-06-05 16:58:08,458 - __main__ - INFO - -----
2023-06-05 16:58:08,458 - __main__ - INFO - PARAMETERS : ---
2023-06-05 16:58:08,458 - __main__ - INFO - n_estimators : 2400
2023-06-05 16:58:08,458 - __main__ - INFO - learning_rate : 0.01
2023-06-05 16:58:08,458 - __main__ - INFO - max_depth : 6
2023-06-05 16:58:08,458 - __main__ - INFO - METRICS : ---
2023-06-05 16:58:08,459 - __main__ - INFO - Accuracy : 0.8044692737430168
2023-06-05 16:58:08,459 - __main__ - INFO - Precision : 0.782608695652174
2023-06-05 16:58:08,459 - __main__ - INFO - Recall : 0.7297297297297297
2023-06-05 16:58:08,459 - __main__ - INFO - F1 : 0.7552447552447553
2023-06-05 16:58:08,459 - __main__ - INFO - -----
2023-06-05 16:58:10,195 - __main__ - WARNING - Start of new run ...
2023-06-05 16:58:12,220 - __main__ - INFO - -----
2023-06-05 16:58:12,221 - __main__ - INFO - PARAMETERS : ---
2023-06-05 16:58:12,221 - __main__ - INFO - n_estimators : 2400
2023-06-05 16:58:12,221 - __main__ - INFO - learning_rate : 0.01
2023-06-05 16:58:12,221 - __main__ - INFO - max_depth : 7
2023-06-05 16:58:12,221 - __main__ - INFO - METRICS : ---
2023-06-05 16:58:12,221 - __main__ - INFO - Accuracy : 0.776536312849162
2023-06-05 16:58:12,221 - __main__ - INFO - Precision : 0.75
2023-06-05 16:58:12,221 - __main__ - INFO - Recall : 0.6891891891891891
2023-06-05 16:58:12,221 - __main__ - INFO - F1 : 0.7183098591549296
2023-06-05 16:58:12,221 - __main__ - INFO - -----
(base) harshit@Harshits-MacBook-Pro titanic %
```

The custom logs are recoding parameters (n\_estimators, learning rate, max depth) and metrics (Accuracy, Precision, recall, F1) successfully.

6. Run the model for different values of hyperparameters – MLFlow keeps a log of hyperparameters and metrics for each run – and then run MLFlow UI:



The screenshot shows the MLFlow UI interface. The top navigation bar includes the MLFlow logo (2.3.2), 'Experiments', and 'Models' tabs. On the right, there are links for 'GitHub' and 'Docs', and a 'Share' button. The main content area is titled 'Experiments' and shows a search bar with 'Default' selected. Below the search bar, there's a 'Description' section. The main table displays a list of runs with columns for 'Run Name', 'Created', 'Duration', 'Source', 'Metrics' (Accuracy, F1, Precision, Recall), and 'Parameters' (learning\_rate, max\_depth, n\_estimators). The table is sorted by Accuracy in descending order. The first run, 'carefree-goose-63', has the highest accuracy of 0.827. The table shows 253 matching runs.

Run Name	Created	Duration	Source	Accuracy	F1	Precision	Recall	learning_rate	max_depth	n_estimators
carefree-goose-63	1 hour ago	29ms	titanic_c...	0.827	0.786	0.803	0.77	0.1	5	400
grandiose-ram-438	1 hour ago	38ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	2400
bald-newt-277	1 hour ago	42ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	2400
valuable-calf-654	1 hour ago	41ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	2300
chill-sheep-678	1 hour ago	41ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	2300
agreeable-worm-12	1 hour ago	40ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	2200
gregarious-ape-649	1 hour ago	40ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	2200
suave-frog-311	1 hour ago	39ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	2100
respected-crab-450	1 hour ago	39ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	2100
luxuriant-wasp-645	1 hour ago	40ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	2000
merciful-dolphin-54	1 hour ago	40ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	2000
vaunted-dove-173	1 hour ago	39ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	1900
industrious-calf-658	1 hour ago	39ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	1900
welcoming-rat-984	1 hour ago	39ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	1800
upset-hog-912	1 hour ago	36ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	1800
polite-roo-812	1 hour ago	37ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	1700
big-deer-610	1 hour ago	38ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	1700
bold-rat-210	1 hour ago	37ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	1600
beautiful-roo-80	1 hour ago	35ms	titanic_c...	0.821	0.778	0.8	0.757	0.1	5	1600
able-dove-195	1 hour ago	34ms	titanic_c...	0.821	0.778	0.8	0.757	0.01	5	1500

After running the for 250+ combinations of hyperparameters, we can see that the best accuracy is achieved with the following hyperparameters:

Learning rate: 0.1  
Max depth: 5  
Estimators: 400

Accuracy: 82.7%  
F1: 0.786  
Precision: 0.803  
Recall: 0.77