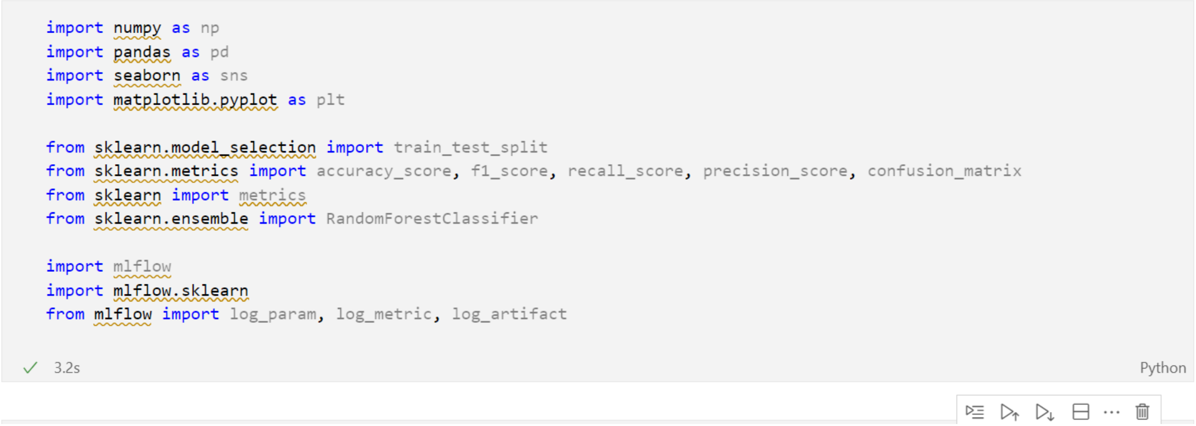
MLflow Steps Screenshots for Titanic Survived Predictions

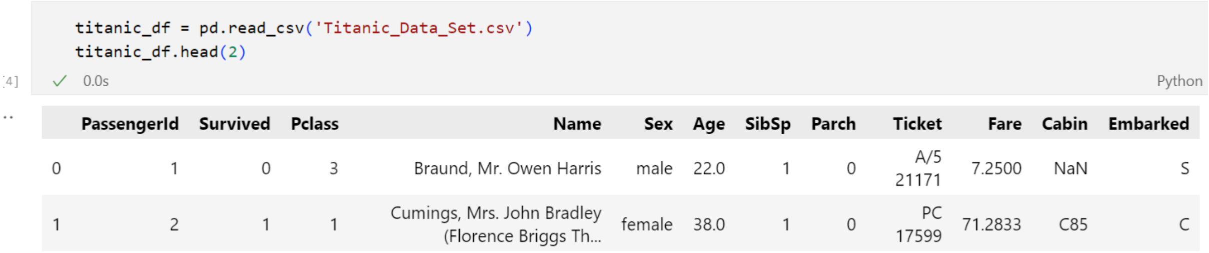
Step-1: Installing the Libraries



```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier

import mlflow
import mlflow.sklearn
from mlflow import log_param, log_metric, log_artifact
```

✓ 3.2s Python

Step-2: Loading Data, EDA, Classification Model Building

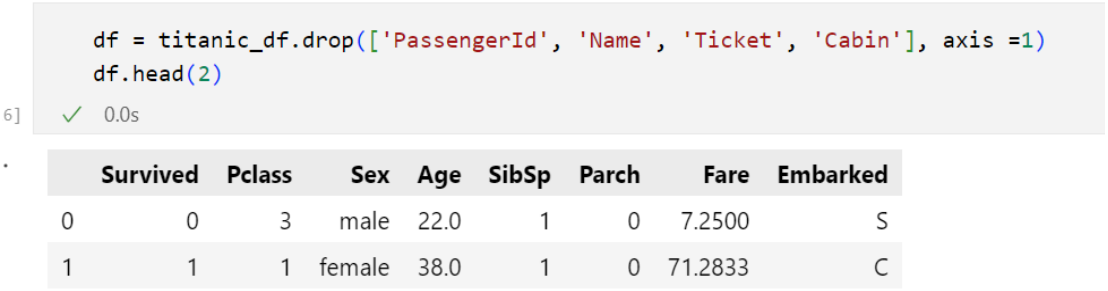


```
titanic_df = pd.read_csv('Titanic_Data_Set.csv')
titanic_df.head(2)
```

✓ 0.0s Python

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

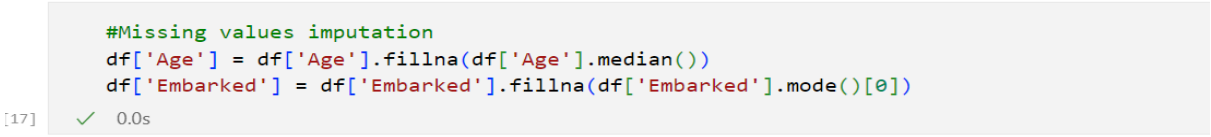
Dropping the redundant columns



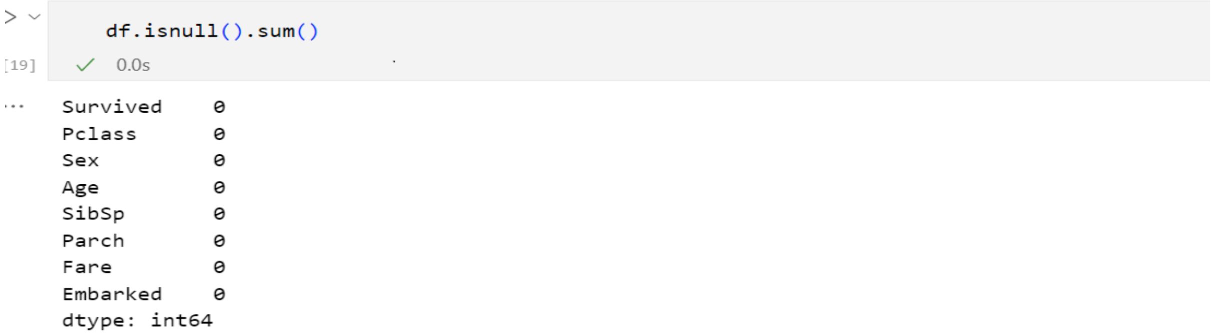
```
df = titanic_df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis =1)
df.head(2)
```

✓ 0.0s

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C

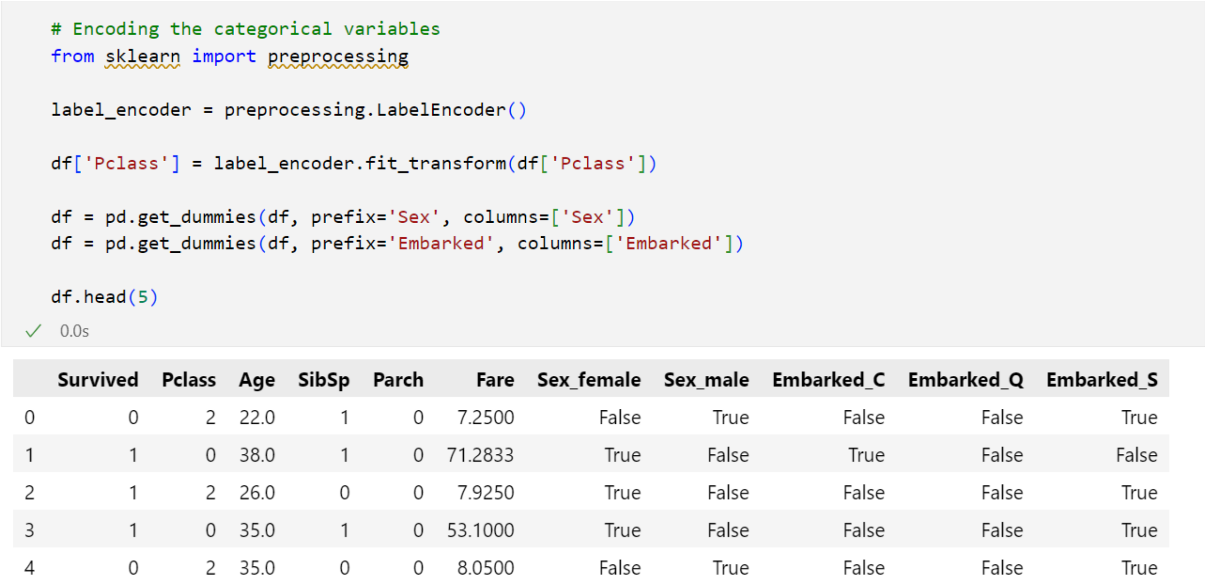


```
#Missing values imputation
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```



```
df.isnull().sum()
```

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

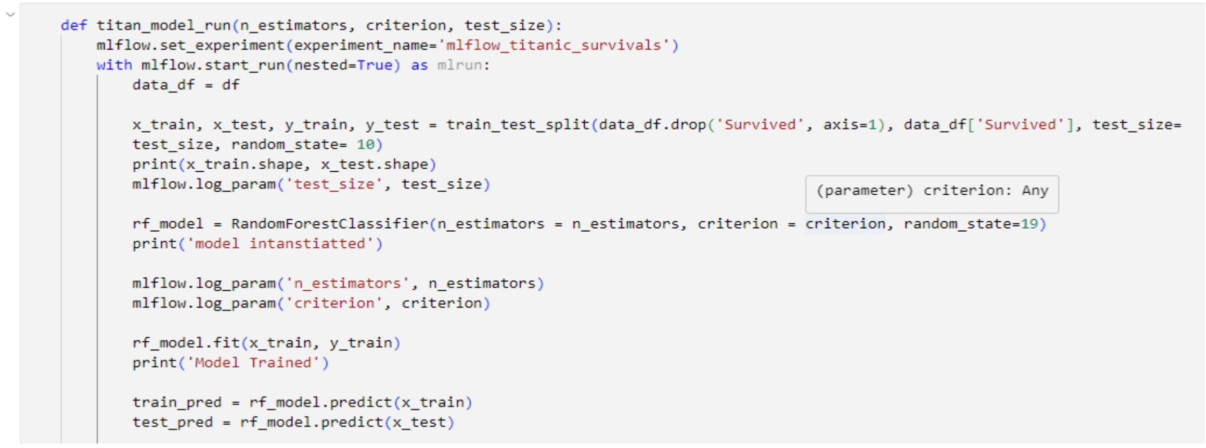


```
# Encoding the categorical variables
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

df['Pclass'] = label_encoder.fit_transform(df['Pclass'])

df = pd.get_dummies(df, prefix='Sex', columns=['Sex'])
df = pd.get_dummies(df, prefix='Embarked', columns=['Embarked'])

df.head(5)
```

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	0	2	22.0	1	0	7.2500	False	True	False	False	True
1	1	0	38.0	1	0	71.2833	True	False	True	False	False
2	1	2	26.0	0	0	7.9250	True	False	False	False	True
3	1	0	35.0	1	0	53.1000	True	False	False	False	True
4	0	2	35.0	0	0	8.0500	False	True	False	False	True

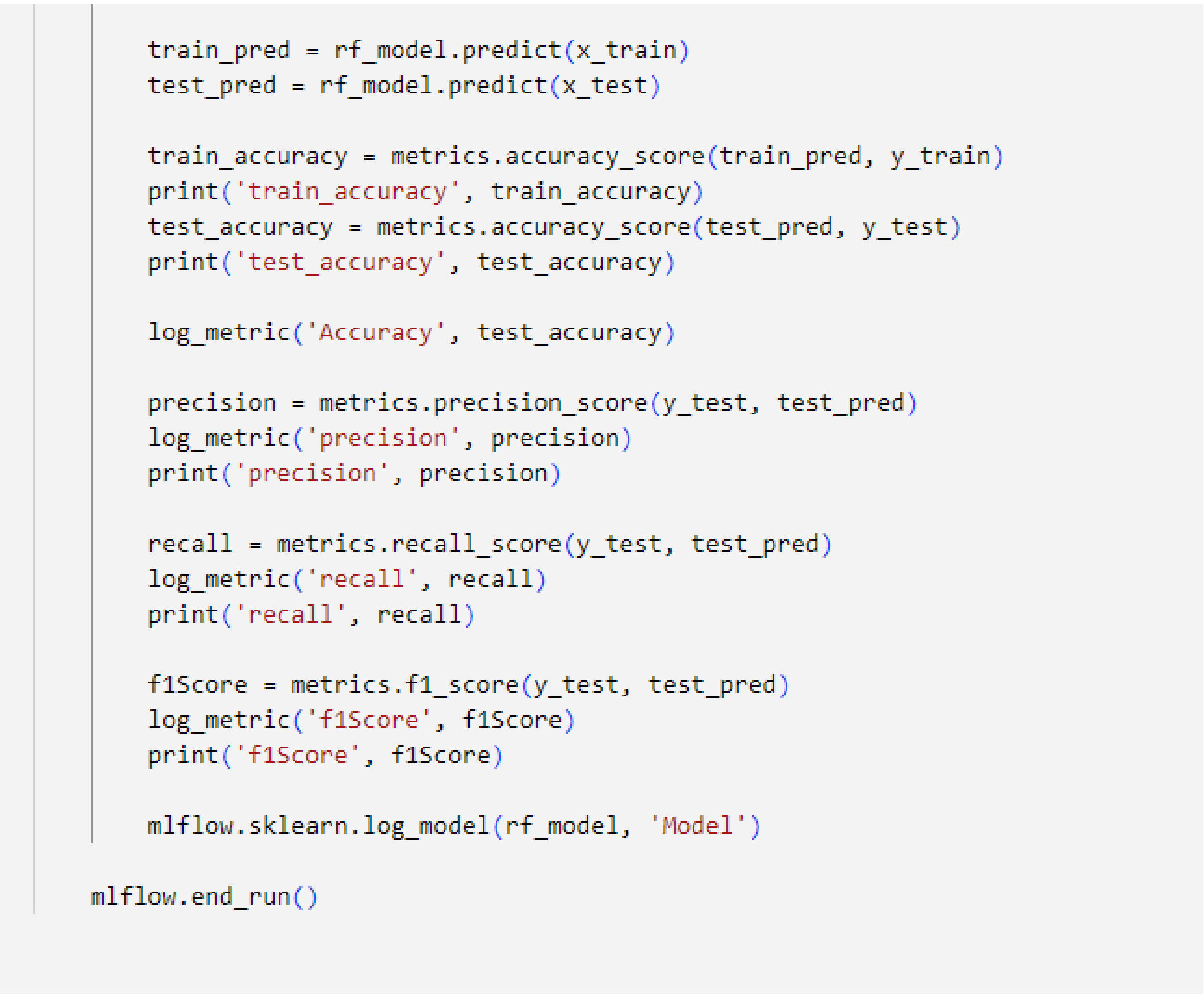
Building the model and predicting the survivors

Logging the parameters, evaluation metrics



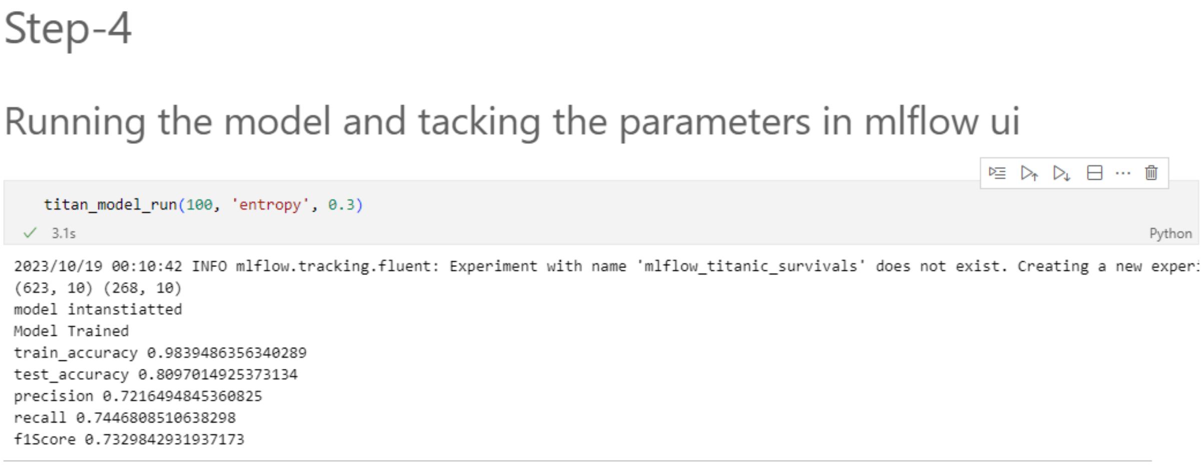
```
def titan_model_run(n_estimators, criterion, test_size):
    mlflow.set_experiment(experiment_name='mlflow_titanic_survivals')
    with mlflow.start_run(nested=True) as mlrun:
        data_df = df

        x_train, x_test, y_train, y_test = train_test_split(data_df.drop('Survived', axis=1), data_df['Survived'], test_size=
        test_size, random_state= 10)
        print(x_train.shape, x_test.shape)
        mlflow.log_param('test_size', test_size)

        rf_model = RandomForestClassifier(n_estimators = n_estimators, criterion = criterion, random_state=19)
        print('model intanstiatted')

        mlflow.log_param('n_estimators', n_estimators)
        mlflow.log_param('criterion', criterion)

        rf_model.fit(x_train, y_train)
        print('Model Trained')

        train_pred = rf_model.predict(x_train)
        test_pred = rf_model.predict(x_test)
```

Step-3: Logging the parameters(previous screenshot) and metrics



```
        train_pred = rf_model.predict(x_train)
        test_pred = rf_model.predict(x_test)

        train_accuracy = metrics.accuracy_score(train_pred, y_train)
        print('train_accuracy', train_accuracy)
        test_accuracy = metrics.accuracy_score(test_pred, y_test)
        print('test_accuracy', test_accuracy)

        log_metric('Accuracy', test_accuracy)

        precision = metrics.precision_score(y_test, test_pred)
        log_metric('precision', precision)
        print('precision', precision)

        recall = metrics.recall_score(y_test, test_pred)
        log_metric('recall', recall)
        print('recall', recall)

        f1Score = metrics.f1_score(y_test, test_pred)
        log_metric('f1Score', f1Score)
        print('f1Score', f1Score)

        mlflow.sklearn.log_model(rf_model, 'Model')

    mlflow.end_run()
```

Step-4: Running the Model and tracking parameters in mlflow ui

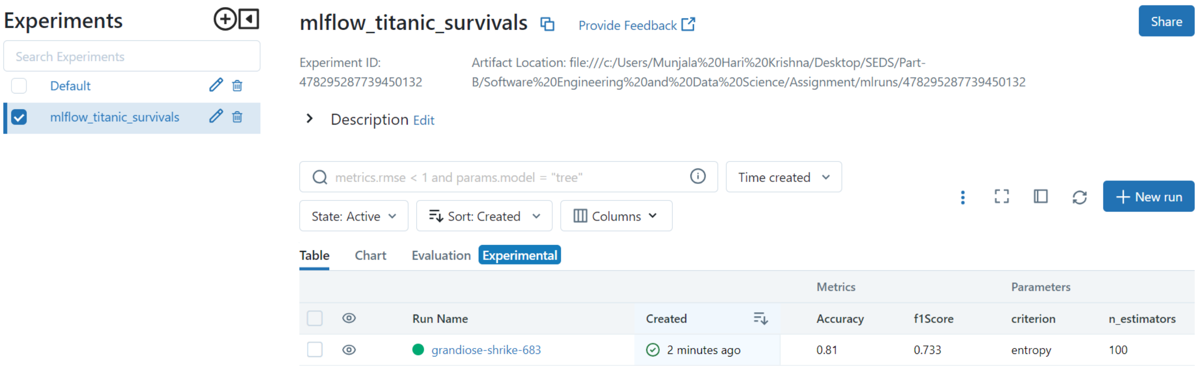


Step-4

Running the model and tacking the parameters in mlflow ui

```
titan_model_run(100, 'entropy', 0.3)
```

```
2023/10/19 00:10:42 INFO mlflow.tracking.fluent: Experiment with name 'mlflow_titanic_survivals' does not exist. Creating a new exper
(623, 10) (268, 10)
model intanstiatted
Model Trained
train_accuracy 0.9839486356340289
test_accuracy 0.8097014925373134
precision 0.7216494845360825
recall 0.7446808510638298
f1Score 0.7329842931937173
```

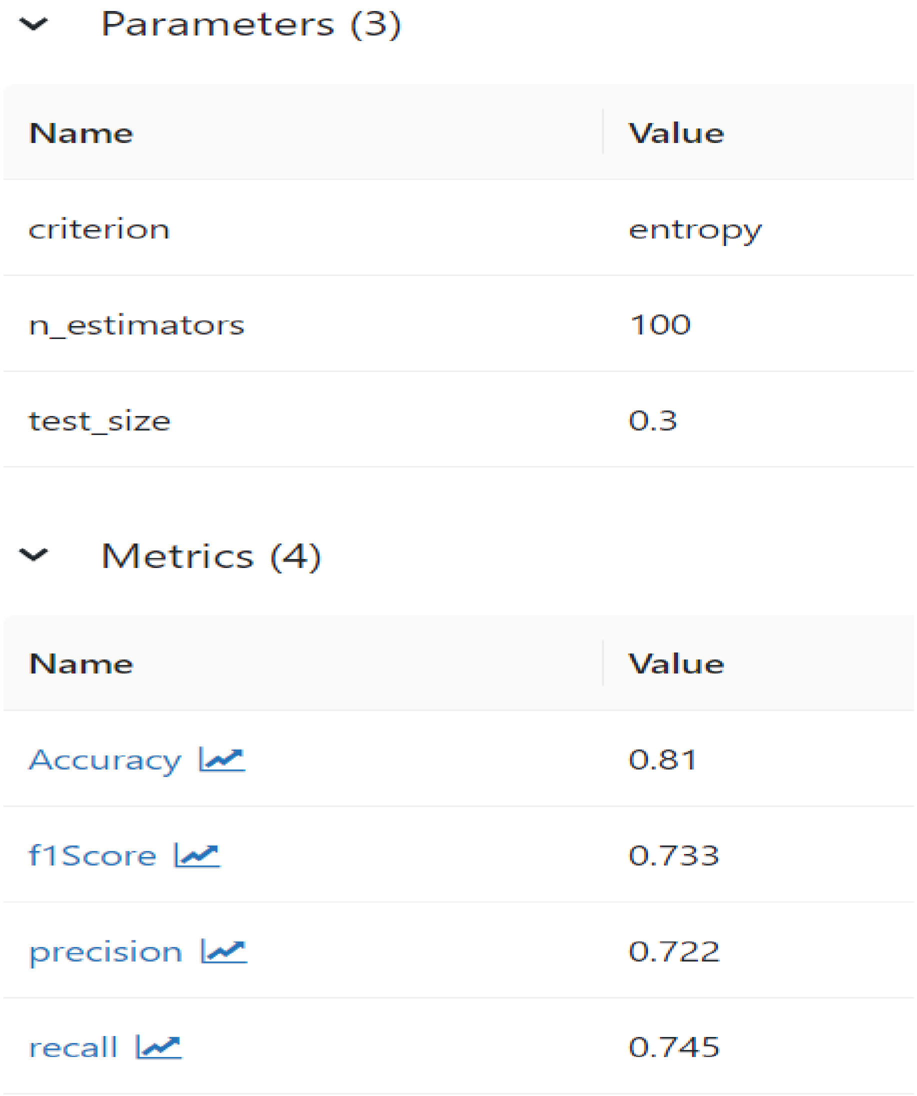


mlflow_titanic_survivals

Experiment ID: 478295287739450132

Artifact Location: file:///c:/Users/Munjala%20Hari%20Krishna/Desktop/SEDS/Part-B/Software%20Engineering%20and%20Data%20Science/Assignment/mlruns/478295287739450132

			Metrics		Parameters	
	Run Name	Created	Accuracy	f1Score	criterion	n_estimators
☐	grandiose-shrike-683	2 minutes ago	0.81	0.733	entropy	100



Parameters (3)

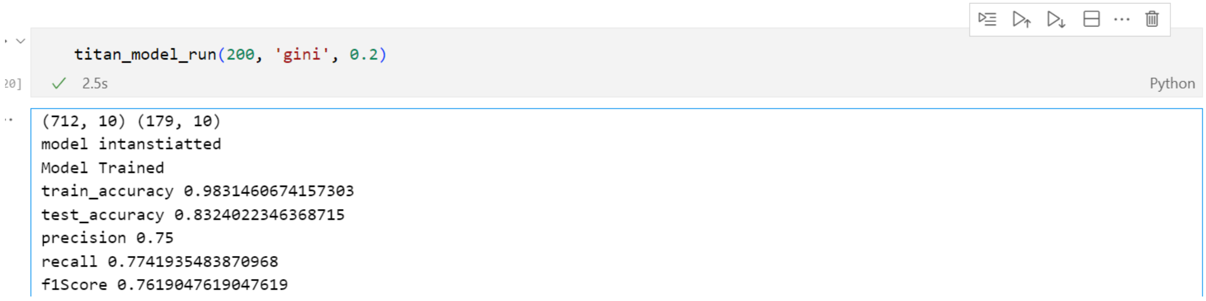
Name	Value
criterion	entropy
n_estimators	100
test_size	0.3

Metrics (4)

Name	Value
Accuracy	0.81
f1Score	0.733
precision	0.722
recall	0.745

Step-5: Re-Running the Model and with different parameters

Re-running the model with different parameter values

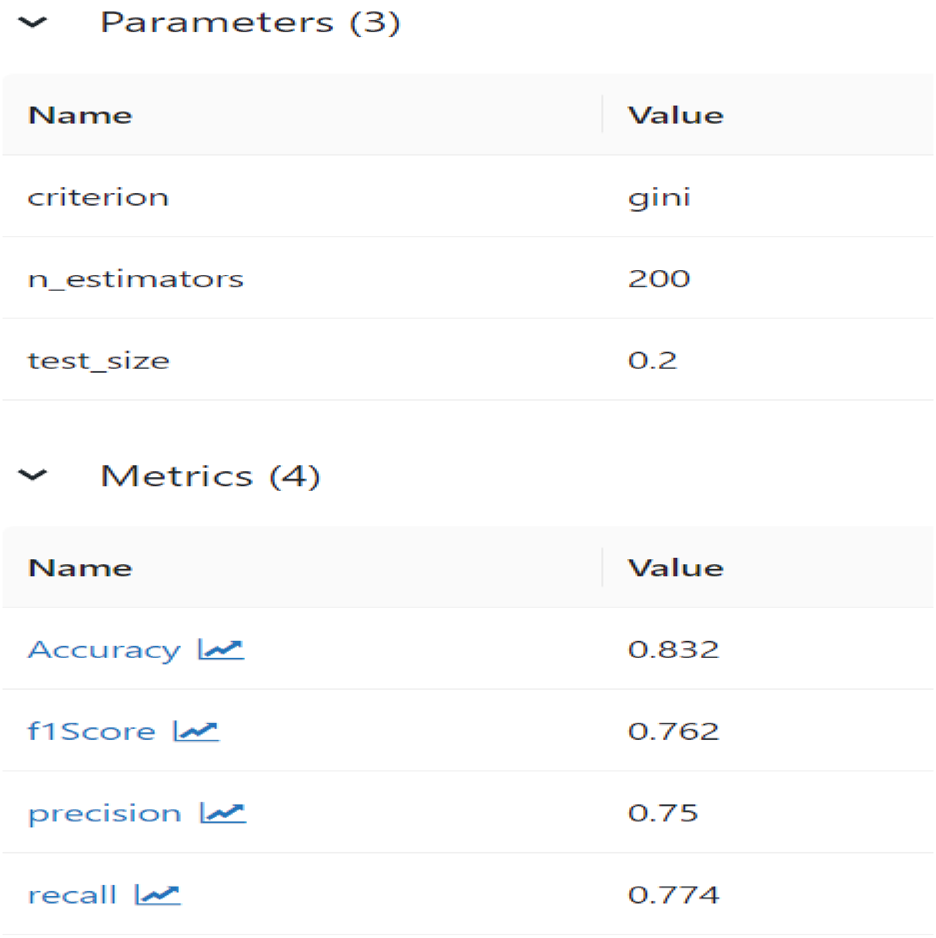


```
titan_model_run(200, 'gini', 0.2)
```

[20] ✓ 2.5s Python

```
(712, 10) (179, 10)
model intanstiatted
Model Trained
train_accuracy 0.9831460674157303
test_accuracy 0.8324022346368715
precision 0.75
recall 0.7741935483870968
f1Score 0.7619047619047619
```

Step-6: Tracking the logged parameters of re-run and comparing it with first run in mlflow ui

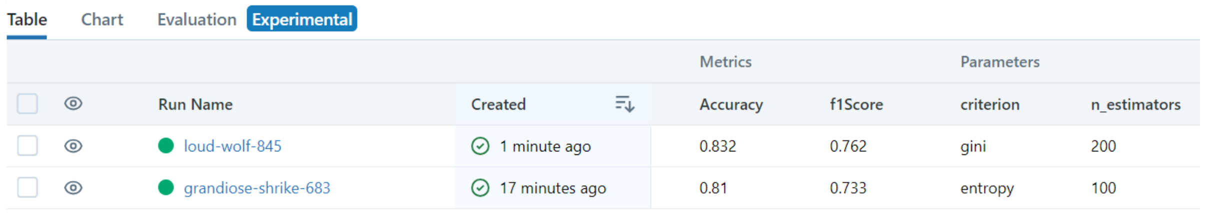


Parameters (3)

Name	Value
criterion	gini
n_estimators	200
test_size	0.2

Metrics (4)

Name	Value
Accuracy	0.832
f1Score	0.762
precision	0.75
recall	0.774



				Metrics		Parameters	
□	👁	Run Name	Created	Accuracy	f1Score	criterion	n_estimators
□	👁	● loud-wolf-845	✓ 1 minute ago	0.832	0.762	gini	200
□	👁	● grandiose-shrike-683	✓ 17 minutes ago	0.81	0.733	entropy	100