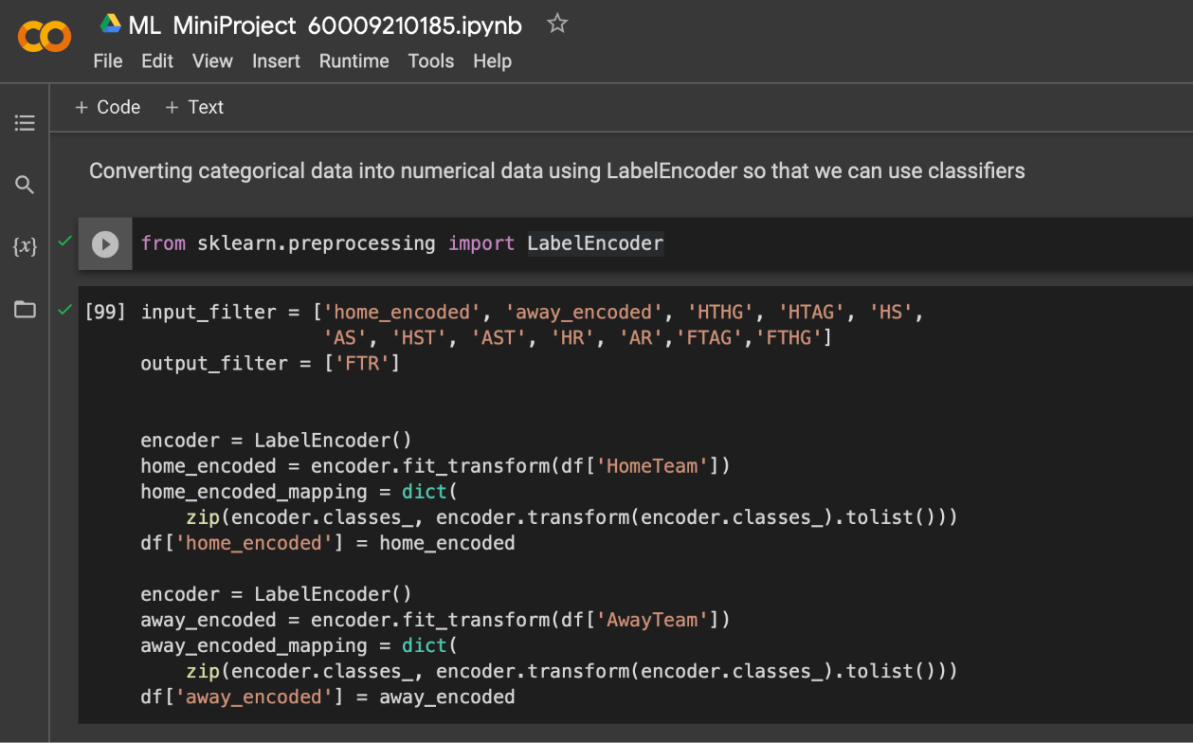


# Task 3

Name - Vedanta Yadav

SAPID - 60009210185

Batch - B/B2



ML MiniProject 60009210185.ipynb ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

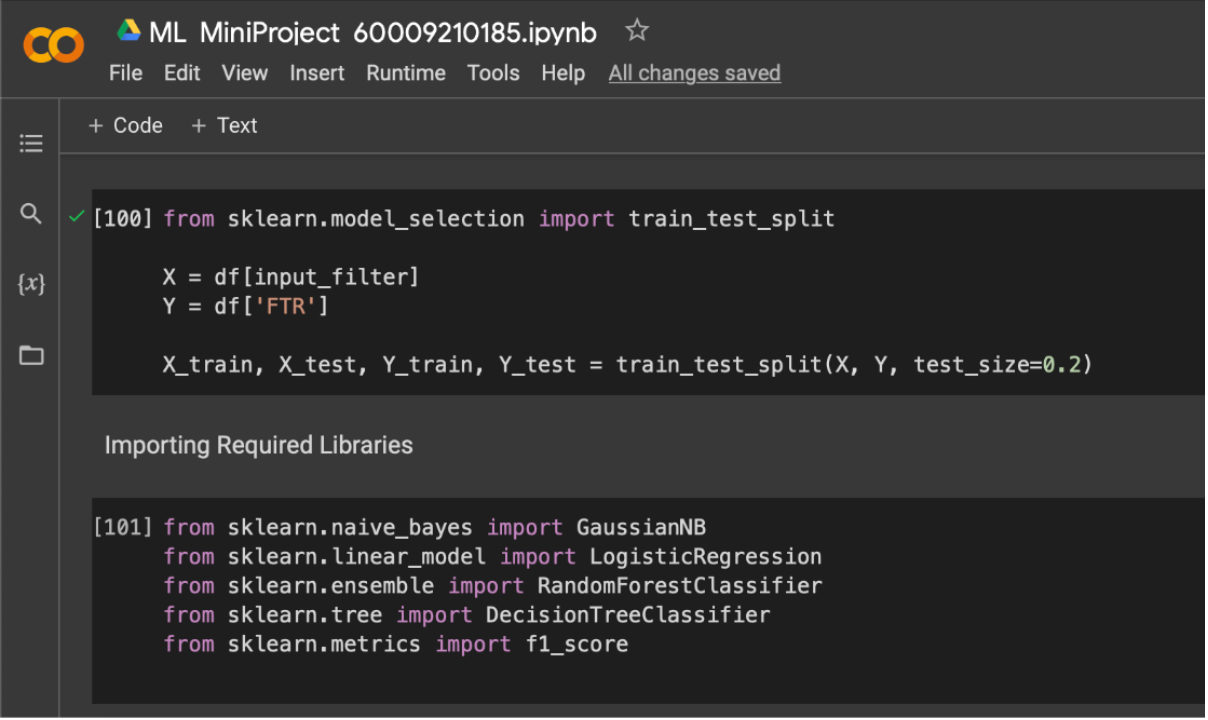
Converting categorical data into numerical data using LabelEncoder so that we can use classifiers

```
[99] from sklearn.preprocessing import LabelEncoder

input_filter = ['home_encoded', 'away_encoded', 'HTHG', 'HTAG', 'HS',
                'AS', 'HST', 'AST', 'HR', 'AR', 'FTAG', 'FTHG']
output_filter = ['FTR']

encoder = LabelEncoder()
home_encoded = encoder.fit_transform(df['HomeTeam'])
home_encoded_mapping = dict(
    zip(encoder.classes_, encoder.transform(encoder.classes_).tolist()))
df['home_encoded'] = home_encoded

encoder = LabelEncoder()
away_encoded = encoder.fit_transform(df['AwayTeam'])
away_encoded_mapping = dict(
    zip(encoder.classes_, encoder.transform(encoder.classes_).tolist()))
df['away_encoded'] = away_encoded
```



ML MiniProject 60009210185.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

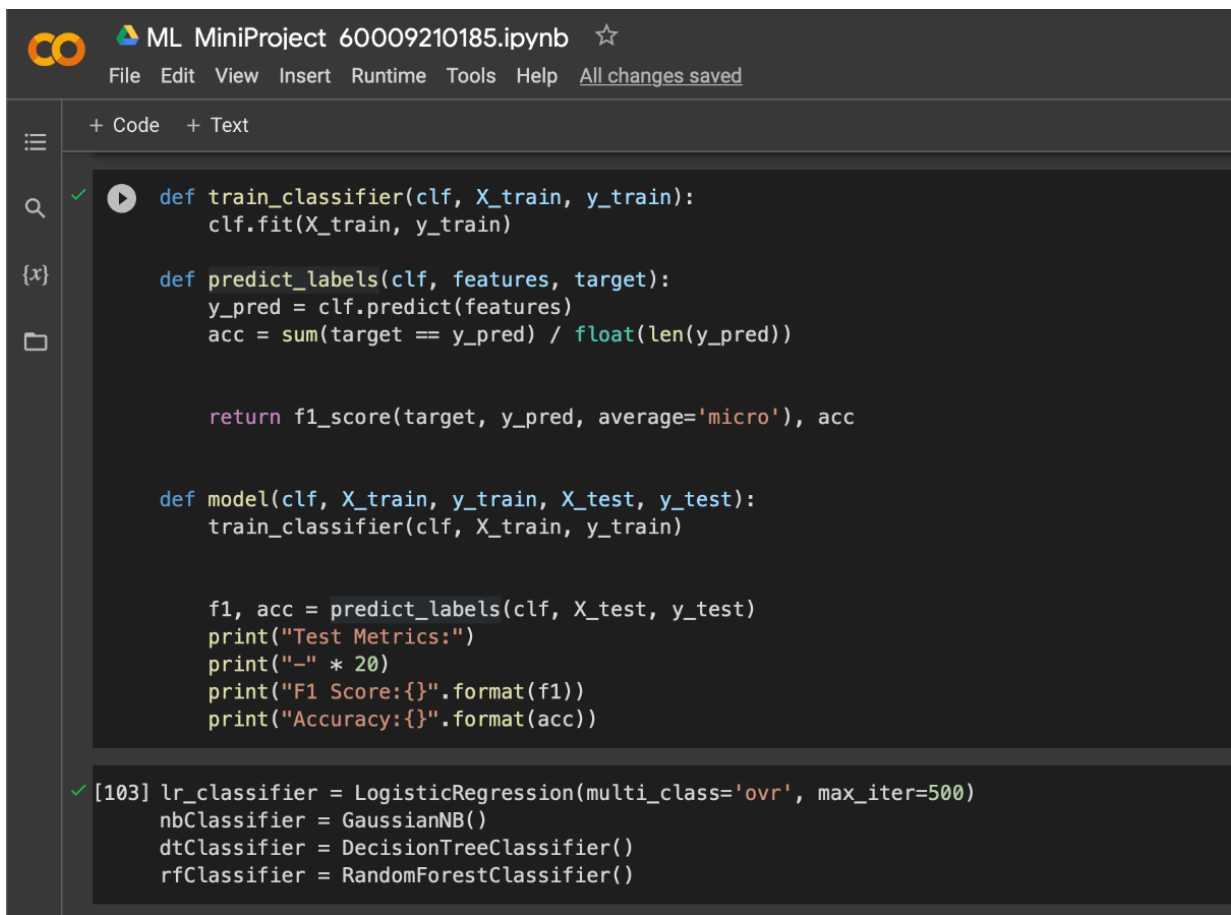
```
[100] from sklearn.model_selection import train_test_split

X = df[input_filter]
Y = df['FTR']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

Importing Required Libraries

```
[101] from sklearn.naive_bayes import GaussianNB
      from sklearn.linear_model import LogisticRegression
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.metrics import f1_score
```



ML MiniProject 60009210185.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def train_classifier(clf, X_train, y_train):
    clf.fit(X_train, y_train)

def predict_labels(clf, features, target):
    y_pred = clf.predict(features)
    acc = sum(target == y_pred) / float(len(y_pred))

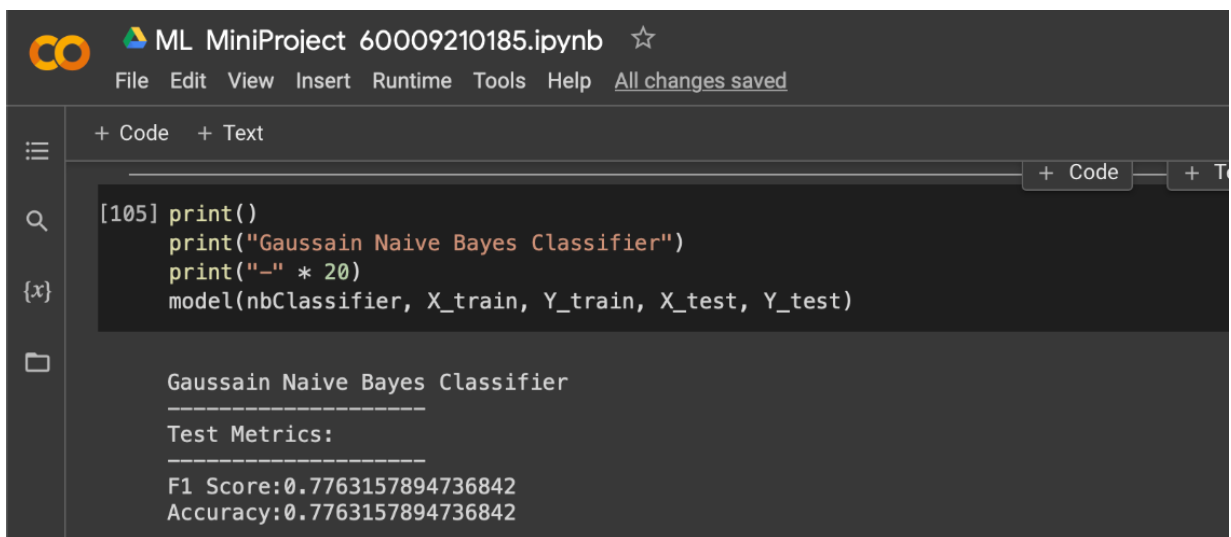
    return f1_score(target, y_pred, average='micro'), acc

def model(clf, X_train, y_train, X_test, y_test):
    train_classifier(clf, X_train, y_train)

    f1, acc = predict_labels(clf, X_test, y_test)
    print("Test Metrics:")
    print("-" * 20)
    print("F1 Score:{}".format(f1))
    print("Accuracy:{}".format(acc))

[103] lr_classifier = LogisticRegression(multi_class='ovr', max_iter=500)
      nbClassifier = GaussianNB()
      dtClassifier = DecisionTreeClassifier()
      rfClassifier = RandomForestClassifier()
```

## Gaussian Naive Bayes



ML MiniProject 60009210185.ipynb ☆

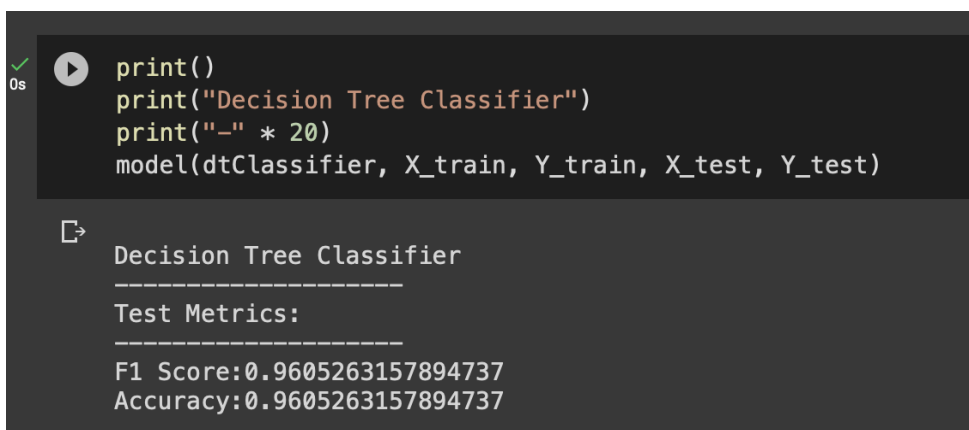
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[105] print()
      print("Gaussain Naive Bayes Classifier")
      print("-" * 20)
      model(nbClassifier, X_train, Y_train, X_test, Y_test)
```

Gaussain Naive Bayes Classifier  
-----  
Test Metrics:  
-----  
F1 Score:0.7763157894736842  
Accuracy:0.7763157894736842

## Decision Tree




0s

```
print()
print("Decision Tree Classifier")
print("-" * 20)
model(dtClassifier, X_train, Y_train, X_test, Y_test)
```


Decision Tree Classifier  
-----  
Test Metrics:  
-----  
F1 Score:0.9605263157894737  
Accuracy:0.9605263157894737

## Random Forest

 ML MiniProject 60009210185.ipynb ☆

File Edit View Insert Runtime Tools Help

+ Code + Text



```
print()
print("Random Forest Classifier")
print("-" * 20)
model(rfClassifier, X_train, Y_train, X_test, Y_test)
```

Random Forest Classifier  
-----  
Test Metrics:  
-----  
F1 Score:0.9868421052631579  
Accuracy:0.9868421052631579

Since from testing different models we conclude that Random Forest Classifier is the best prediction model for this dataset, as it gave the most accuracy