Lead Scoring Assignment X Education

By
Pankaj Kumar
Tejesh N

Problem Statement

- An education company named X Education sells online courses to industry professionals. On any given day, many professionals who are interested in the courses land on their website and browse for courses.
- The company markets its courses on several websites and search engines like Google. Once these people land on the website, they might browse the courses or fill up a form for the course or watch some videos. When these people fill up a form providing their email address or phone number, they are classified to be a lead. Moreover, the company also gets leads through past referrals. Once these leads are acquired, employees from the sales team start making calls, writing emails, etc. Through this process, some of the leads get converted while most do not. The typical lead conversion rate at X education is around 30%.

Problem Statement continued

- Now, although X Education gets a lot of leads, its lead conversion rate is very poor. For example, if, say, they acquire 100 leads in a day, only about 30 of them are converted. To make this process more efficient, the company wishes to identify the most potential leads, also known as ‘Hot Leads’. If they successfully identify this set of leads, the lead conversion rate should go up as the sales team will now be focusing more on communicating with the potential leads rather than making calls to everyone.
- The company requires you to build a model wherein you need to assign a lead score to each of the leads such that the customers with higher lead score have a higher conversion chance and the customers with lower lead score have a lower conversion chance. The CEO, in particular, has given a ballpark of the target lead conversion rate to be around 80%.

Solution Approach

- We have performed Data cleaning, EDA and Data Preparation on provided data set.
- We understood that this is a Logistic Regression problem as the target variable “Converted” is a Binary/categorical output.
- We then performed Logistic Regression Modelling and come up with Probability of Hot Leads.
- We found out the lead score of all the Leads.

Observations and Findings

- **Initial data set have volume**

Number of records: 9240

Number of features: 37

- **Provided data set have Lead conversion percentage as**

Not Converted	61.461039
Converted	38.538961

- **Some features have value as "Selected" which in Business term is null values. Which we converted in null values and then depending on %of null values in that feature either we dropped it or imputed missing value**

Observations and Findings

- **Few categorical feature have very less value of a particular kind compared to other value. Which we then grouped them appropriately** for example Country had India in large volume, other country has very less % we grouped them all as "Other"
- **After completing our data cleaning and EDA process below are the stats of dataset**

Dataset volume Before and After

Number of records Before: 9240

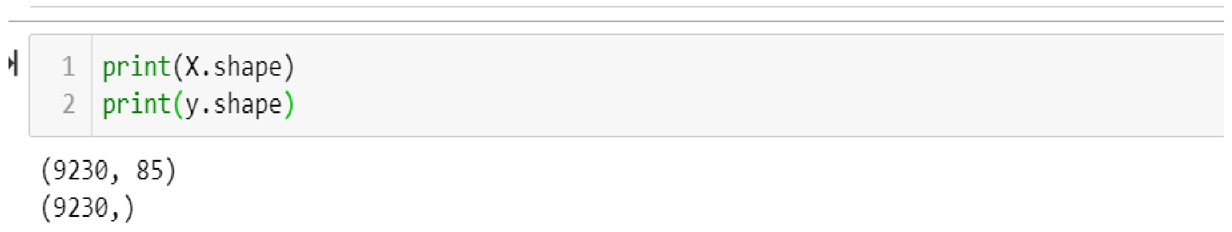
Number of features Before: 37

Number of records After: 9230

Number of features After: 28

Data Preparation

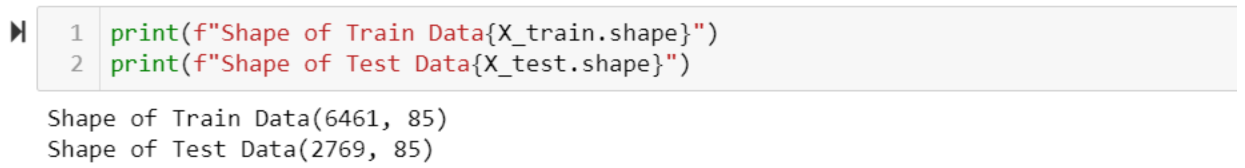
- We first created dummy variable for categorical features
- Created X and y Independent and Target variable datframes



```
print(X.shape)
print(y.shape)
```

```
(9230, 85)
(9230,)
```

- Split the data into train and test in 70:30 keeping random_state and stratify



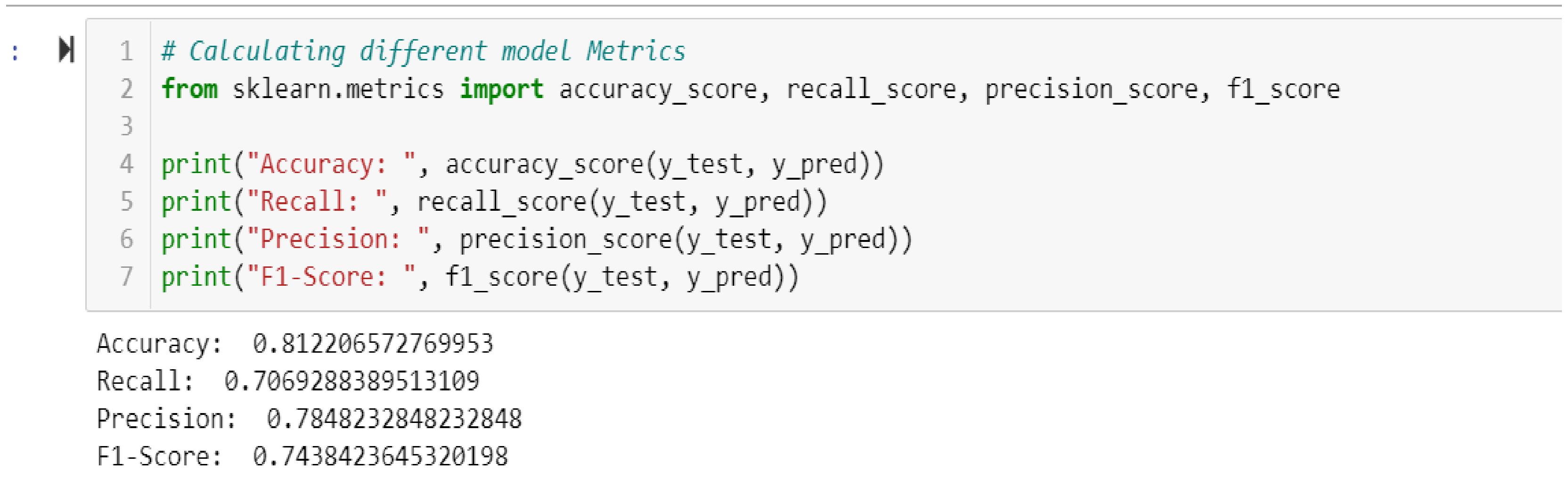
```
print(f"Shape of Train Data{X_train.shape}")
print(f"Shape of Test Data{X_test.shape}")
```

```
Shape of Train Data(6461, 85)
Shape of Test Data(2769, 85)
```

- Performed Scaling of numerical feature using StandardScaling

Modelling

- Created 1st Logistic Regression Model using all the features
- Below are Metrics on test data. Here default cut-off of 0.5 is applied



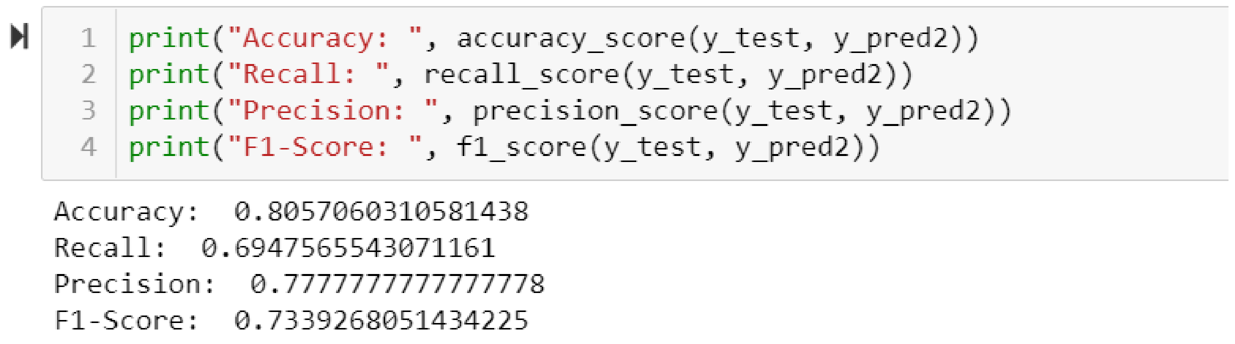
```
# Calculating different model Metrics
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score

print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Recall: ", recall_score(y_test, y_pred))
print("Precision: ", precision_score(y_test, y_pred))
print("F1-Score: ", f1_score(y_test, y_pred))
```

```
Accuracy:  0.812206572769953
Recall:  0.7069288389513109
Precision:  0.7848232848232848
F1-Score:  0.7438423645320198
```

Modelling

- After 1st model, we optimised the model. We used **Recursive Feature Elimination (RFE)** to select top 15 features.
- Then we also performed **Variance Inflation Factor (VIF)** on these selected 15 features and found VIF score is in acceptable range.
- **Model Metrics after RFE and VIF**

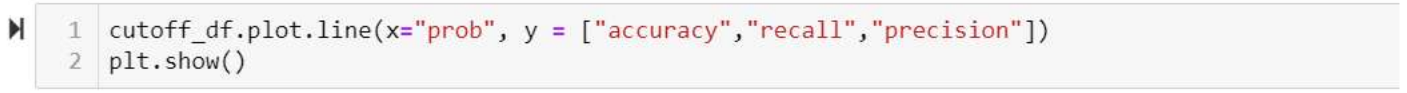


```python
print("Accuracy: ", accuracy_score(y_test, y_pred2))
print("Recall: ", recall_score(y_test, y_pred2))
print("Precision: ", precision_score(y_test, y_pred2))
print("F1-Score: ", f1_score(y_test, y_pred2))
```

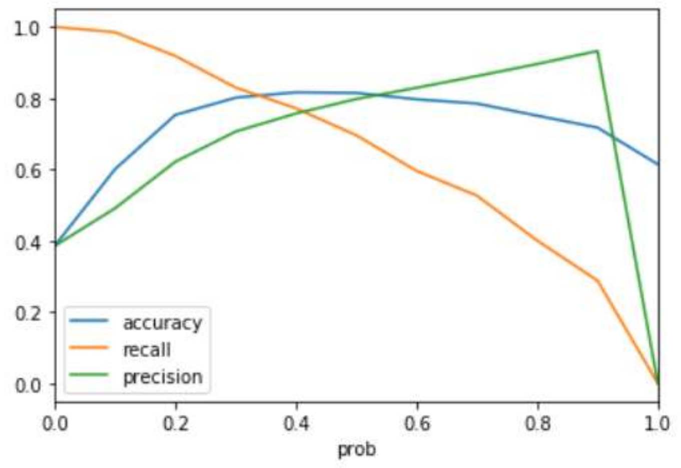
```
Accuracy:  0.8057060310581438
Recall:  0.6947565543071161
Precision:  0.7777777777777778
F1-Score:  0.7339268051434225
```

Modelling Cut-Off tuning

- As previous models were on cut-off of 0.5
- We then performed cut-off tunning to find optimal point between Recall and Precision



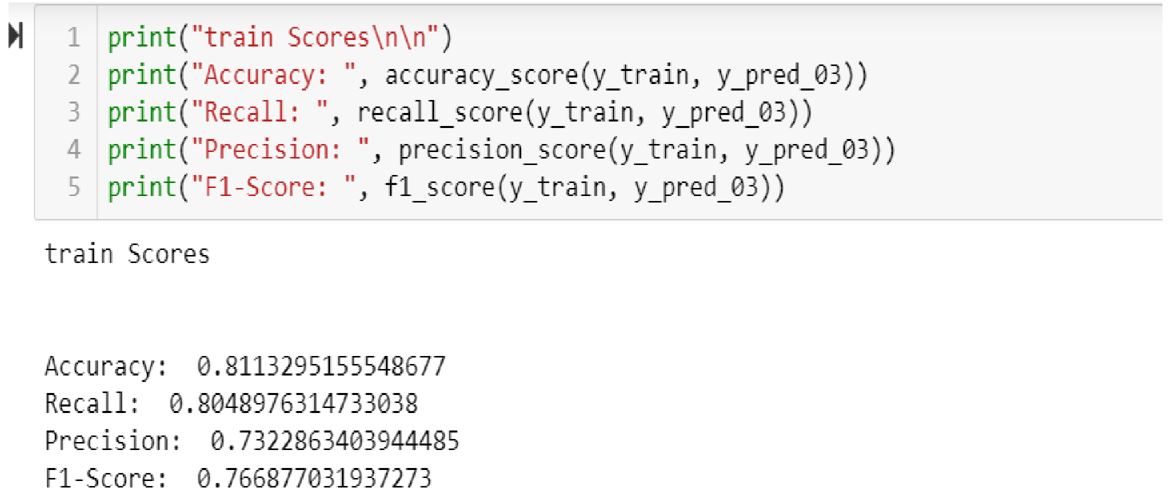
```
cutoff_df.plot.line(x="prob", y = ["accuracy","recall","precision"])
plt.show()
```



- **Optimal cut-off point seems to be at 0.35**

Final Model using cut-off point 0.35

- We chose optimal cut-off point as 0.35 for our final model.
- Model Metrics for train dataset for cut-off point as 0.35



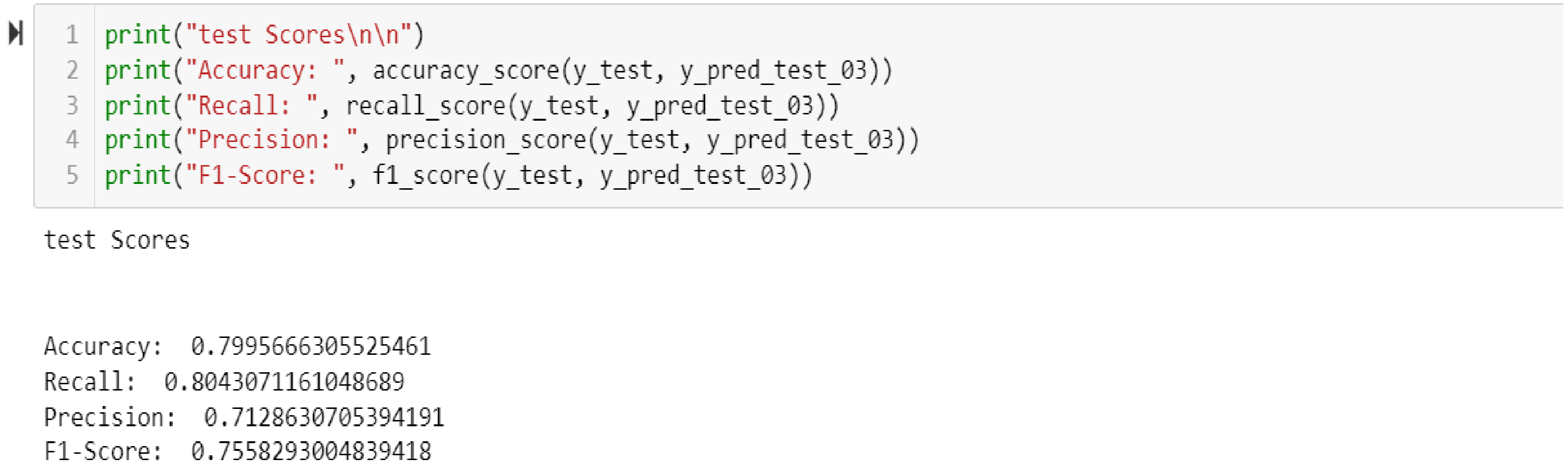
```
print("train Scores\n\n")
print("Accuracy: ", accuracy_score(y_train, y_pred_03))
print("Recall: ", recall_score(y_train, y_pred_03))
print("Precision: ", precision_score(y_train, y_pred_03))
print("F1-Score: ", f1_score(y_train, y_pred_03))
```

```
train Scores


Accuracy:  0.8113295155548677
Recall:  0.8048976314733038
Precision:  0.7322863403944485
F1-Score:  0.766877031937273
```

Final Model using cut-off point 0.35

- Model Metrics for test dataset for cut-off point as 0.35



```
print("test Scores\n\n")
print("Accuracy: ", accuracy_score(y_test, y_pred_test_03))
print("Recall: ", recall_score(y_test, y_pred_test_03))
print("Precision: ", precision_score(y_test, y_pred_test_03))
print("F1-Score: ", f1_score(y_test, y_pred_test_03))
```

```
test Scores


Accuracy:  0.7995666305525461
Recall:  0.8043071161048689
Precision:  0.7128630705394191
F1-Score:  0.7558293004839418
```

Calculation of Lead Score

- We finally calculated lead score of Train and Test data and stored them in in dataframes

Thank You