CISC 839
Topics in Data Analytics
Competition 2

Supervised learning

Sahil Shethiya
20186685
19ss55@queensu.ca
School of Computing

Kamal Andani
20188032
19kaa3@queenu.ca
Department of Electrical and
Computer Engineering

Software Packages Used:

1. **Pandas:** Python has long been great for data managing and preparation, but less so for data analysis and modeling. Pandas is an open-source library that helps fill this gap, enabling you to carry out your entire data analysis workflow in Python.
   - ➢ To install type the following in Anaconda prompt (terminal):
     ```
     $ conda install pandas
     ```

2. **scikit-learn:** Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.
   - ➢ To install type the following in Anaconda prompt (terminal):
     ```
     $ conda install scikit-learn
     ```

3. **imbalanced-learn**: It is a python package offering several re-sampling techniques commonly used in datasets showing a strong between-class imbalance. It is compatible with scikit-learn.
   - ➢ To install type the following in Anaconda prompt (terminal):
     ```
     $ conda install  imbalanced-learn
     ```

Description of Analytics Process:

- First of all, there are many missing values in the data in the form of '?' which we replaced with the NaN. Secondly, we found the percentage of missing data and found out of the three columns i.e. **weight**, **medical_speciality**, **payer_code** had more than 40% data missing. Thus, we dropped those columns.
- Secondly, for the columns with only a few missing values, we only removed the record containing missing values in **race**, **diag_1**, **diag_2**, **diag_3.**
- There was only one entry in **gender** containing an unknown value, thus we dropped that record.
- By looking count of unique values in every column, we found out that columns were containing one value or almost one value in every record. Thus, we remove those columns which are examide, citoglipton, metformin-rosiglitazone, acetohexamide, glimepiride-pioglitazone, metformin-pioglitazone.
- Moving forward we converted the categorical columns which are **gender, race, age, change, diabetesMed, readmitted,** and all **features of medication** into numeric ones.
- The columns **diag_1, diag_2, diag_3** contain values having character 'V' or 'E', which we replaced with the value 1000 or 1100 respectively as the values are between 1 to 999 for other **ICD9** code.
- For **training,** we dropped patient_nbr2 and encounter_id2 columns as they are just identification numbers and not useful for classification.
- During analytics, we found that there was imbalanced data that we handle by oversampling using the **SMOTE** technique.
- After oversampling, we scale the numeric predictor variable using **StandardScaler.**
- Then we split the data into train and validation set with a ratio of 70/30 percent.

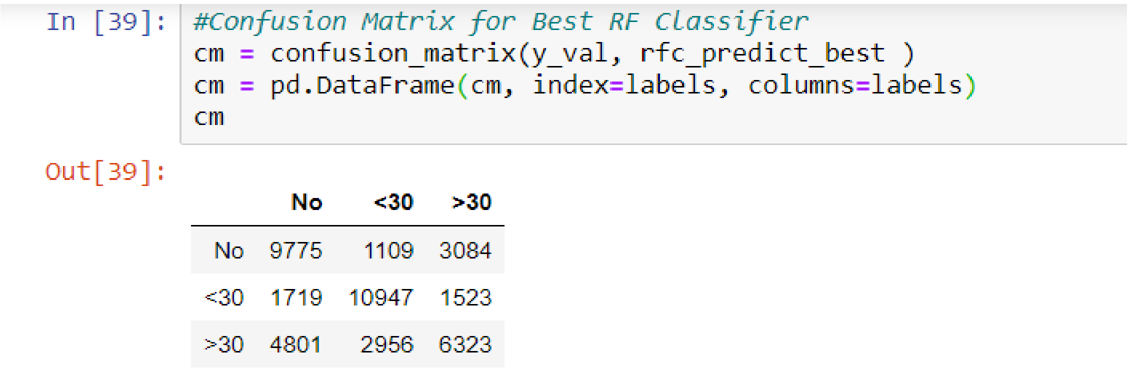
- Firstly for modeling, we used **RandomForest** Classifier with a depth of 25(random) in which we scored the accuracy of 63 percent. After that, we used the **cross-validation** technique to find the best parameter for RandomForest Classifier and scored the accuracy of 66 percent.
- Lastly, we have also used **Binary classification** to predict if the patient will be admitted within 30 days or no, for that we have converted labels where 0 indicates if the patient will be admitted to the hospital within 30 days and 1 if the patient will be admitted to hospital after 30 days or won't be admitted.

Other Approaches tried:

- We tried decision tree classifiers but the model gave us an accuracy of 52% on multiclass classification problems and 80% for binary classification problems.

Evaluation:

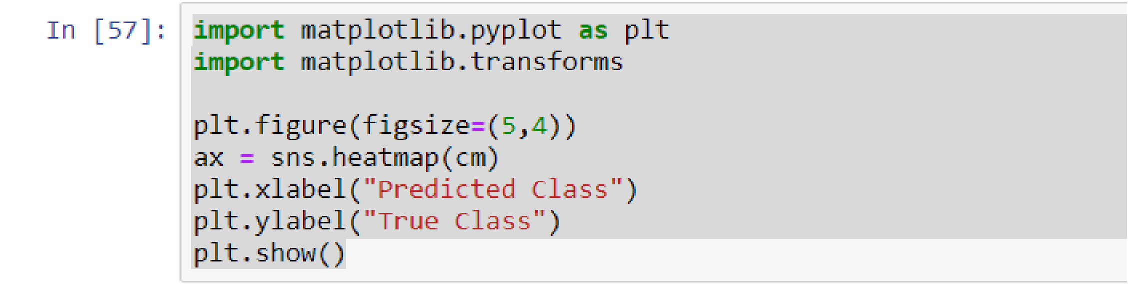
Confusion matrix on validation data for multiclass classification problem:



```
In [39]: #Confusion Matrix for Best RF Classifier
         cm = confusion_matrix(y_val, rfc_predict_best )
         cm = pd.DataFrame(cm, index=labels, columns=labels)
         cm
```

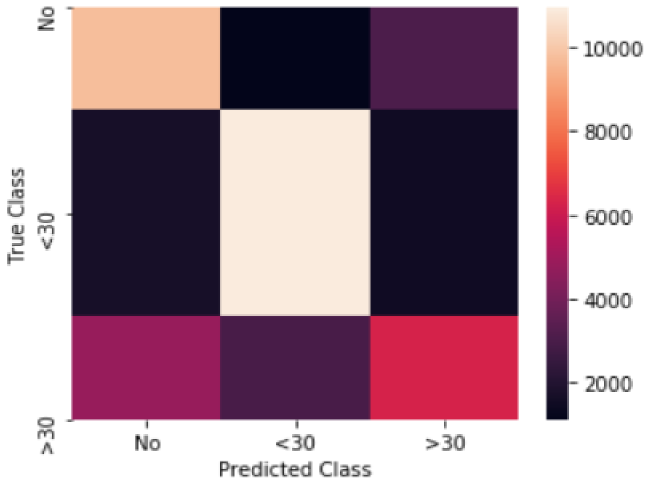
Out[39]:

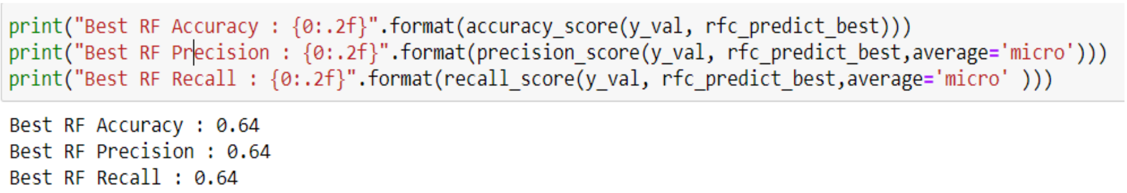
	No	<30	>30
No	9775	1109	3084
<30	1719	10947	1523
>30	4801	2956	6323



```
In [57]: import matplotlib.pyplot as plt
         import matplotlib.transforms

         plt.figure(figsize=(5,4))
         ax = sns.heatmap(cm)
         plt.xlabel("Predicted Class")
         plt.ylabel("True Class")
         plt.show()
```

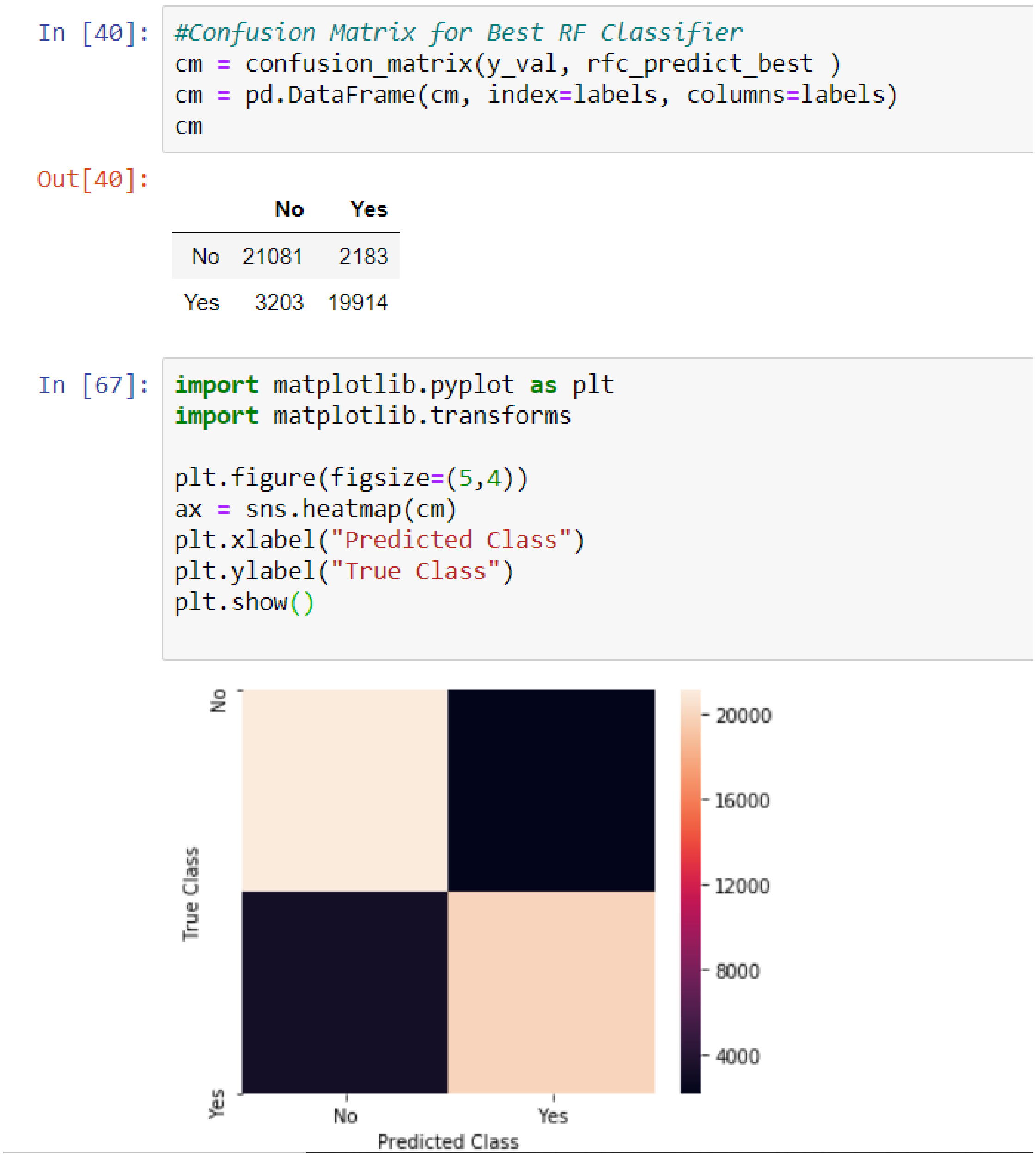




```
print("Best RF Accuracy : {0:.2f}".format(accuracy_score(y_val, rfc_predict_best)))
print("Best RF Precision : {0:.2f}".format(precision_score(y_val, rfc_predict_best,average='micro')))
print("Best RF Recall : {0:.2f}".format(recall_score(y_val, rfc_predict_best,average='micro' )))
```

```
Best RF Accuracy : 0.64
Best RF Precision : 0.64
Best RF Recall : 0.64
```

Confusion matrix on validation data for binary classification problem:



```
In [40]: #Confusion Matrix for Best RF Classifier
         cm = confusion_matrix(y_val, rfc_predict_best )
         cm = pd.DataFrame(cm, index=labels, columns=labels)
         cm
```

Out[40]:

	No	Yes
No	21081	2183
Yes	3203	19914

```
In [67]: import matplotlib.pyplot as plt
         import matplotlib.transforms

         plt.figure(figsize=(5,4))
         ax = sns.heatmap(cm)
         plt.xlabel("Predicted Class")
         plt.ylabel("True Class")
         plt.show()
```