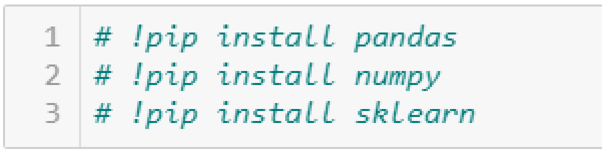
User guide for Customer Retention Model

<u>Prerequisites</u>

1) Python 3
2) Jupyter Notebook
3) Visual Studio Code
4) GitHub Account

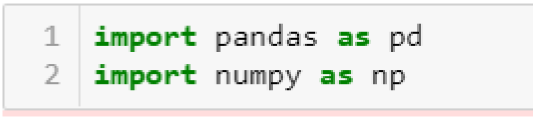
<u>Section 1: Data Cleaning</u>

1) Open Jupyter notebook
2) Open Data Cleaning.ipynb
3) Uncomment and run the first cell to install pandas, numpy, sklearn.



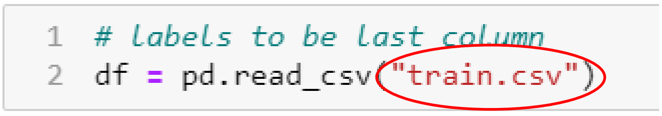
```
# !pip install pandas
# !pip install numpy
# !pip install sklearn
```

4) Run the 2nd cell to import pandas and numpy packages



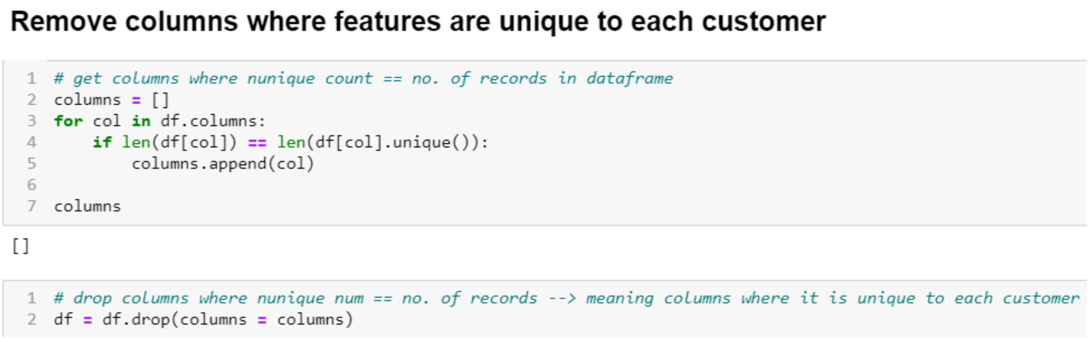
```
import pandas as pd
import numpy as np
```

5) Import the dataset to be cleaned. Change "train.csv" to the name of the file. This can be both the training file and testing file.
****Note that for training file, labels need to be the at the right most column and column name must be 'labels'**



```
# labels to be last column
df = pd.read_csv("train.csv")
```

6) Continue running the cells to check how many records there are in the dataset, number of rows that contain NA values and drop these NA values.
7) Check for features that are unique to each customer using df.nunique()
    a. If there are features that are unique to each customer for eg. CustomerID, remove them by running the next 2 cell.



Remove columns where features are unique to each customer

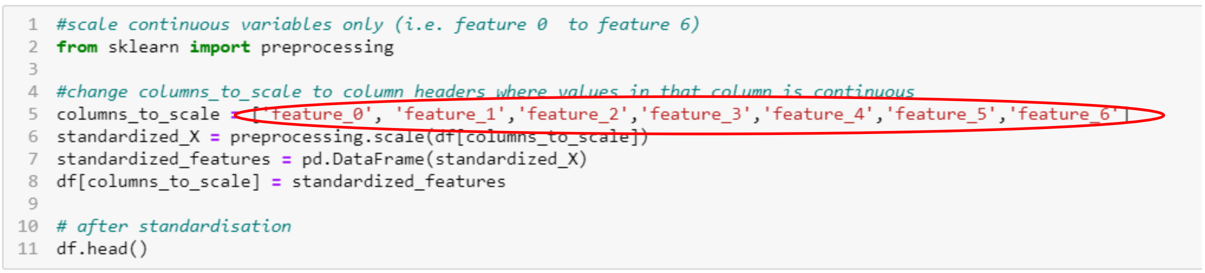
```
# get columns where nunique count == no. of records in dataframe
columns = []
for col in df.columns:
    if len(df[col]) == len(df[col].unique()):
        columns.append(col)

columns
```

```
[]
```

```
# drop columns where nunique num == no. of records --> meaning columns where it is unique to each customer
df = df.drop(columns = columns)
```

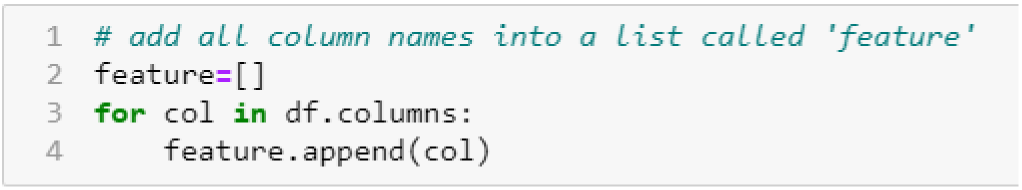
8) Check that columns that have values unique to each customer has been removed by running the next cell, df.head()

9) Next, we will be performing data standardization. Data standardization should only be applied to column where values are **continuous (**decimal for eg. 1.23) NOT discrete (whole number for eg. 0,1,2,3)
   Note: Change columns_to_scale to column headers where its values are continuous



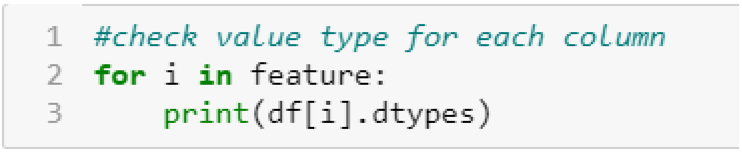
```python
#scale continuous variables only (i.e. feature 0  to feature 6)
from sklearn import preprocessing

#change columns_to_scale to column headers where values in that column is continuous
columns_to_scale = ['feature_0', 'feature_1','feature_2','feature_3','feature_4','feature_5','feature_6']
standardized_X = preprocessing.scale(df[columns_to_scale])
standardized_features = pd.DataFrame(standardized_X)
df[columns_to_scale] = standardized_features

# after standardisation
df.head()
```

10) Next, we will perform label encoding to convert textual data to numerical data as machine learning models cannot work with text.
    a. Add all the columns name into a list called feature



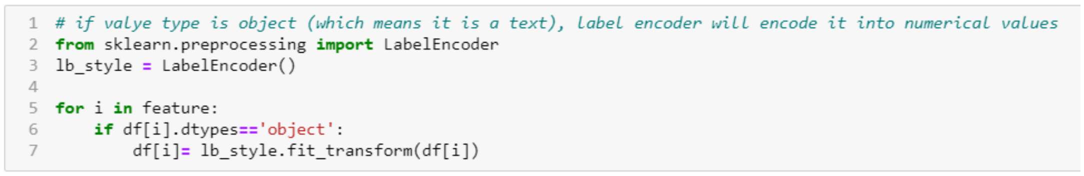
```python
# add all column names into a list called 'feature'
feature=[]
for col in df.columns:
    feature.append(col)
```

    b. Check value type of each column



```python
#check value type for each column
for i in feature:
    print(df[i].dtypes)
```

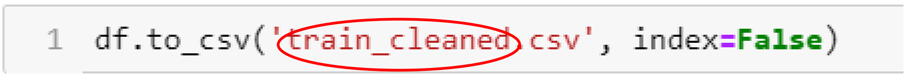
    c. If value type of column is 'object', it will be encoded into numbers



```python
# if valye type is object (which means it is a text), label encoder will encode it into numerical values
from sklearn.preprocessing import LabelEncoder
lb_style = LabelEncoder()

for i in feature:
    if df[i].dtypes=='object':
        df[i]= lb_style.fit_transform(df[i])
```

    d. Check that textual values have been changed into numbers by running df.head()

11) Export the cleaned dataset. You can rename the file to any name you prefer by changing 'train_cleaned'. Note that the extension '.csv' is required.

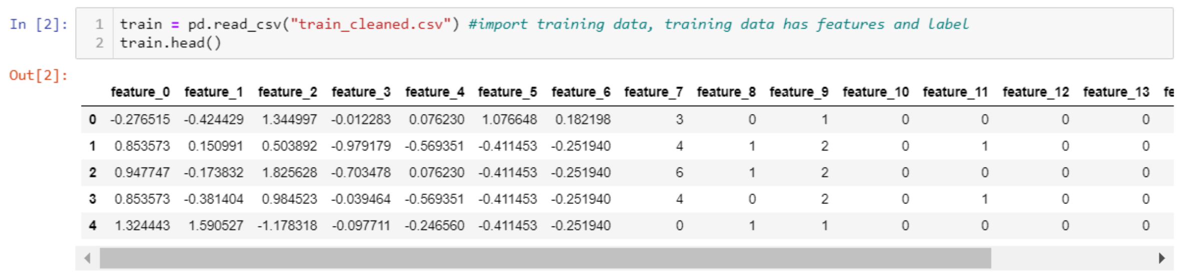


```python
df.to_csv('train_cleaned.csv', index=False)
```

12) Perform data cleaning for both the train and test file and export them so that they can be uploaded into the flask application (Section 3).

Section 2: Model optimization (Get hyperparameters for models)

1) Open Model Optimisation.ipynb
2) Run the first box to import required modules
3) Import only the training file that you have just cleaned. (Change the filename accordingly if you saved it in another name)

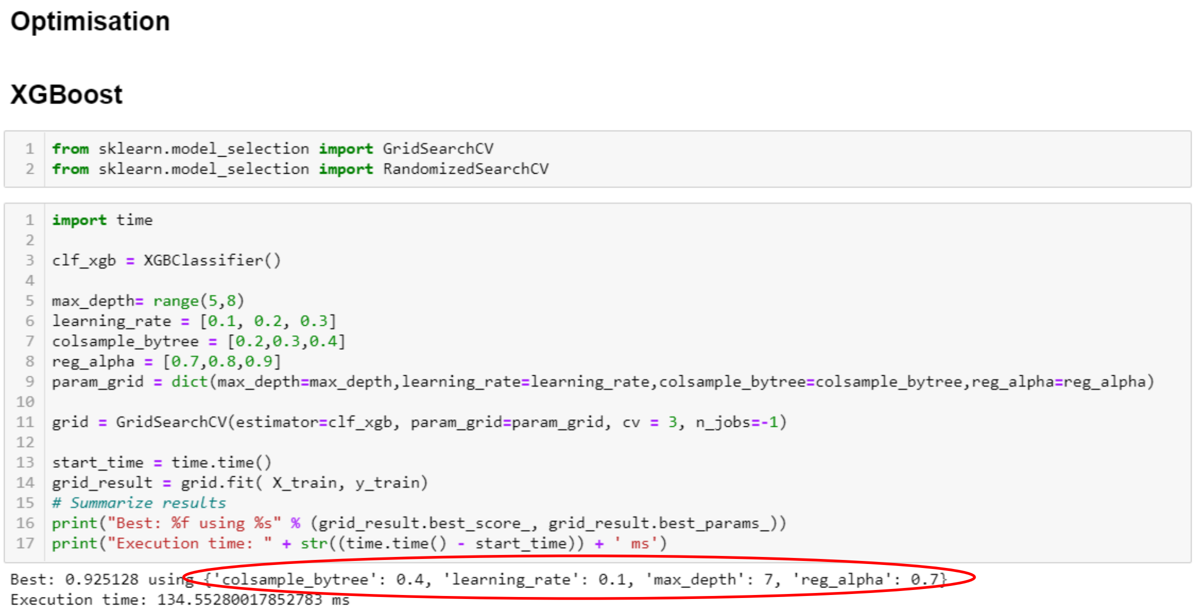


In [2]:
```
train = pd.read_csv("train_cleaned.csv") #import training data, training data has features and label
train.head()
```

Out[2]:

	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	feature_9	feature_10	feature_11	feature_12	feature_13	fe
0	-0.276515	-0.424429	1.344997	-0.012283	0.076230	1.076648	0.182198	3	0	1	0	0	0	0	
1	0.853573	0.150991	0.503892	-0.979179	-0.569351	-0.411453	-0.251940	4	1	2	0	1	0	0	
2	0.947747	-0.173832	1.825628	-0.703478	0.076230	-0.411453	-0.251940	6	1	2	0	0	0	0	
3	0.853573	-0.381404	0.984523	-0.039464	-0.569351	-0.411453	-0.251940	4	0	2	0	1	0	0	
4	1.324443	1.590527	-1.178318	-0.097711	-0.246560	-0.411453	-0.251940	0	1	1	0	0	0	0	

4) Run all the codes below
5) After running all the codes, go to 'Optimisation' section (as shown below).



Optimisation

XGBoost

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

```
import time

clf_xgb = XGBClassifier()

max_depth= range(5,8)
learning_rate = [0.1, 0.2, 0.3]
colsample_bytree = [0.2,0.3,0.4]
reg_alpha = [0.7,0.8,0.9]
param_grid = dict(max_depth=max_depth,learning_rate=learning_rate,colsample_bytree=colsample_bytree,reg_alpha=reg_alpha)

grid = GridSearchCV(estimator=clf_xgb, param_grid=param_grid, cv = 3, n_jobs=-1)

start_time = time.time()
grid_result = grid.fit( X_train, y_train)
# Summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

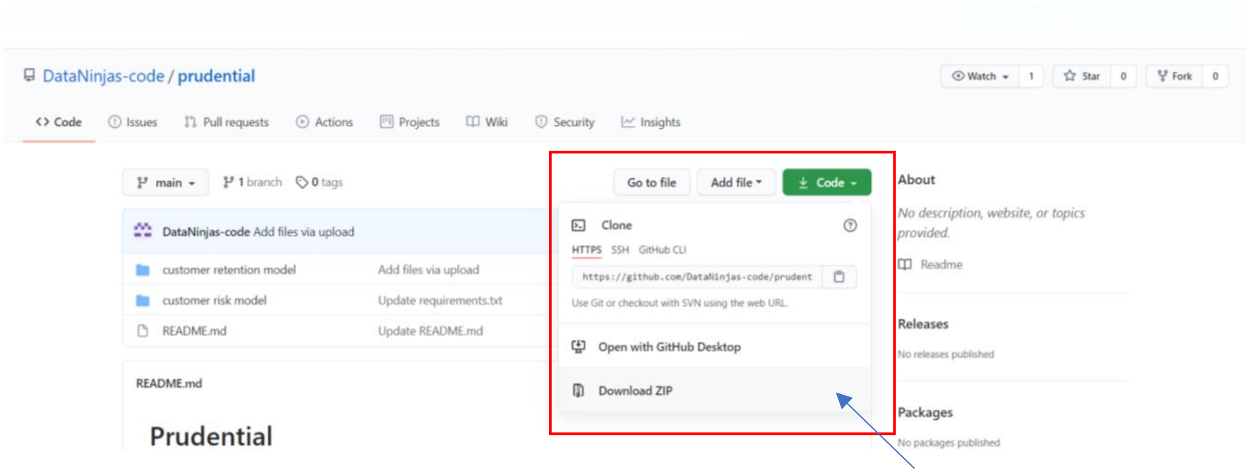
```
Best: 0.925128 using {'colsample_bytree': 0.4, 'learning_rate': 0.1, 'max_depth': 7, 'reg_alpha': 0.7}
Execution time: 134.55280017852783 ms
```

For each of the models, get the best hyperparameters.
For eg. For XGBoost, the best parameters are {'colsample_bytree': 0.4, 'learning_rate': 0.1, 'max_depth': 7, 'reg_alpha': 0.7}

6) Take note of all these best hyperparameters as you need to replace them in the flask application in the next section.

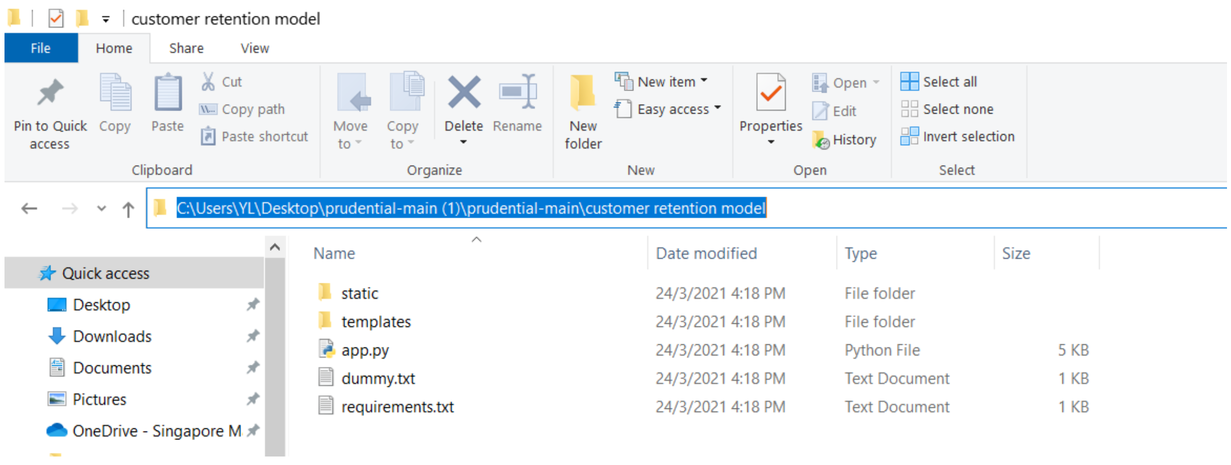
<u>Section 3: Customer Retention Prediction Model</u>

1) Go to https://github.com/DataNinjas-code/prudential
- Download the file



DataNinjas-code / prudential
Code Issues Pull requests Actions Projects Wiki Security Insights
Watch 1 Star 0 Fork 0
main 1 branch 0 tags
Go to file Add file Code
DataNinjas-code Add files via upload
customer retention model Add files via upload
customer risk model Update requirements.txt
README.md Update README.md
Clone
HTTPS SSH GitHub CLI
https://github.com/DataNinjas-code/prudent
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP
README.md
Prudential
About
No description, website, or topics provided.
Readme
Releases
No releases published
Packages
No packages published

- Extract the contents in the zip file
- Click into the folder and into 'customer retention model'
- Click on the file path (make sure your file path is similar to the screenshot)

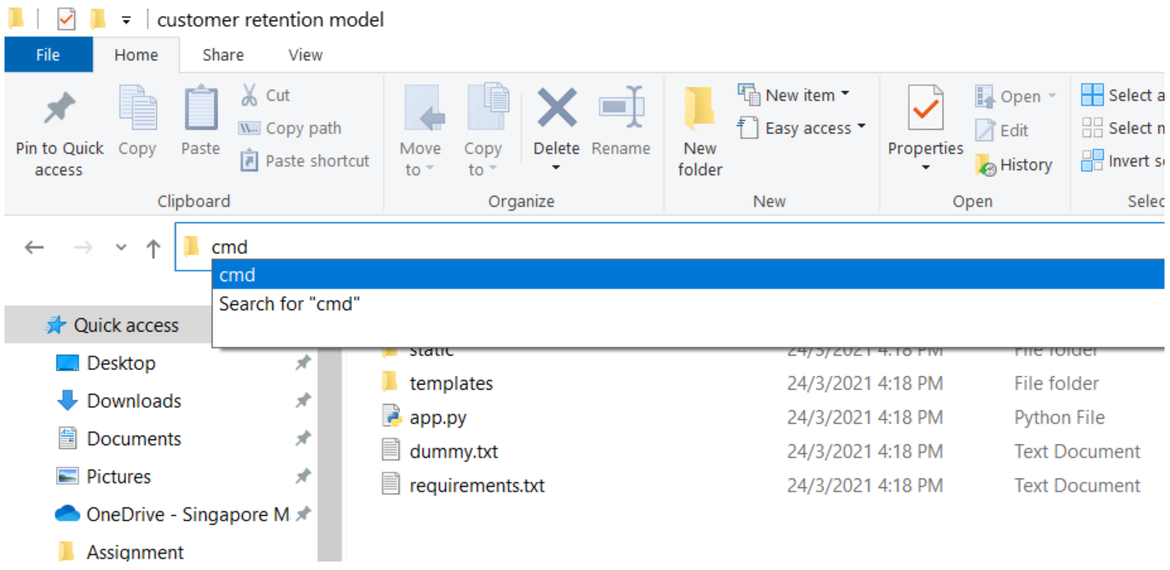


customer retention model
File Home Share View
Pin to Quick access Copy Paste Cut Copy path Paste shortcut
Clipboard
Move to Copy to Delete Rename
Organize
New folder New item Easy access
New
Properties Open Edit History
Open
Select all Select none Invert selection
Select
Quick access
Desktop
Downloads
Documents
Pictures
OneDrive - Singapore M

C:\Users\YL\Desktop\prudential-main (1)\prudential-main\customer retention model

Name	Date modified	Type	Size
static	24/3/2021 4:18 PM	File folder	
templates	24/3/2021 4:18 PM	File folder	
app.py	24/3/2021 4:18 PM	Python File	5 KB
dummy.txt	24/3/2021 4:18 PM	Text Document	1 KB
requirements.txt	24/3/2021 4:18 PM	Text Document	1 KB

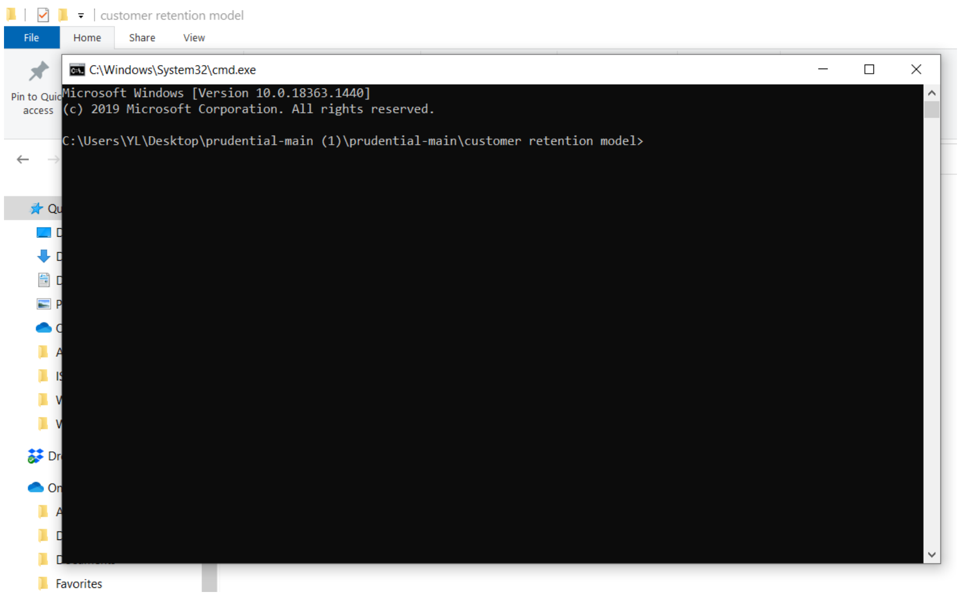
- Type 'cmd' and press enter



customer retention model
File Home Share View
Pin to Quick access Copy Paste Cut Copy path Paste shortcut
Clipboard
Move to Copy to Delete Rename
Organize
New folder New item Easy access
New
Properties Open Edit History
Open
Select a Select n Invert s
Selec
cmd
cmd
Search for "cmd"
Quick access
Desktop
Downloads
Documents
Pictures
OneDrive - Singapore M
Assignment

static	24/3/2021 4:18 PM	File folder
templates	24/3/2021 4:18 PM	File folder
app.py	24/3/2021 4:18 PM	Python File
dummy.txt	24/3/2021 4:18 PM	Text Document
requirements.txt	24/3/2021 4:18 PM	Text Document

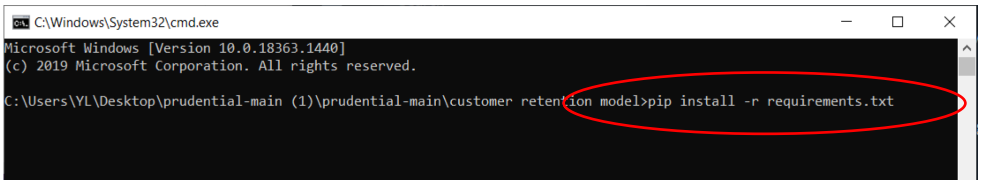
- You will see your command prompt being opened up.



```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\YL\Desktop\prudential-main (1)\prudential-main\customer retention model>
```

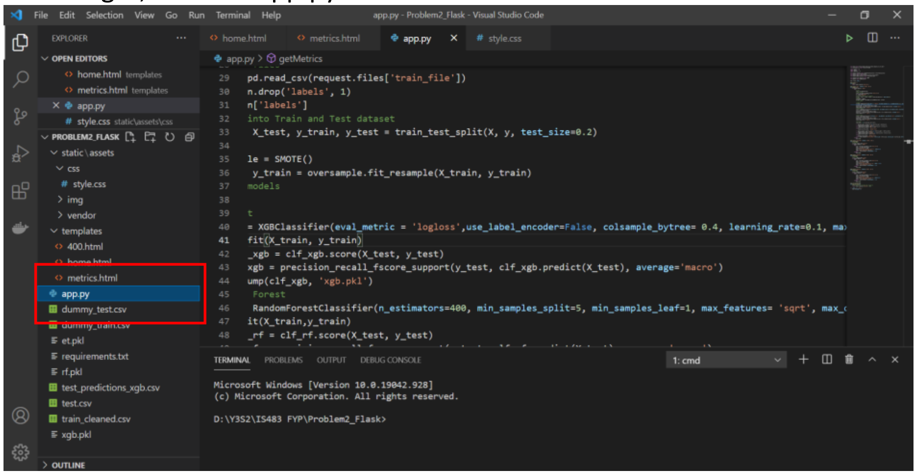
2) Pip install all the required packages using requirements.txt by typing:

```
pip install -r requirements.txt
```



```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\YL\Desktop\prudential-main (1)\prudential-main\customer retention model>pip install -r requirements.txt
```

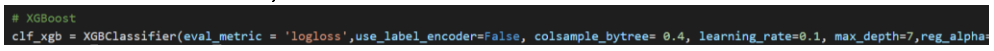
3) Open the flask application in Visual Studio Code or any Software Development Environment. If you are using Visual Studio Code, follow the instructions below:
   a. Open Visual Studio Code
   b. Click on File > Open Folder > Select the ‘customer retention model’ folder
   c. At the right, click on ‘app.py’



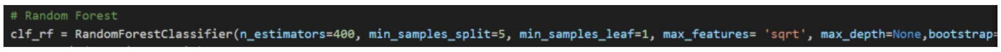
```
pd.read_csv(request.files['train_file'])
n.drop('labels', 1)
n['labels']
into Train and Test dataset
 X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

le = SMOTE()
 y_train = oversample.fit_resample(X_train, y_train)
models


t
= XGBClassifier(eval_metric = 'logloss',use_label_encoder=False, colsample_bytree= 0.4, learning_rate=0.1, ma
fit(X_train, y_train)
_xgb = clf_xgb.score(X_test, y_test)
xgb = precision_recall_fscore_support(y_test, clf_xgb.predict(X_test), average='macro')
ump(clf_xgb, 'xgb.pkl')
 Forest
 RandomForestClassifier(n_estimators=400, min_samples_split=5, min_samples_leaf=1, max_features= 'sqrt', max_
it(X_train,y_train)
_rf = clf_rf.score(X_test, y_test)
```

```
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

D:\Y3S2\IS483 FYP\Problem2_Flask>
```

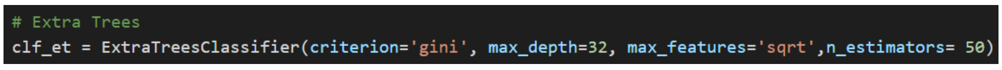
d. Inside app.py, change the hyperparameters of the 3 models (XGBoost, Random Forest and Extra Trees) which was derived in section 2.



```
# XGBoost
clf_xgb = XGBClassifier(eval_metric = 'logloss',use_label_encoder=False, colsample_bytree= 0.4, learning_rate=0.1, max_depth=7,reg_alpha=
```



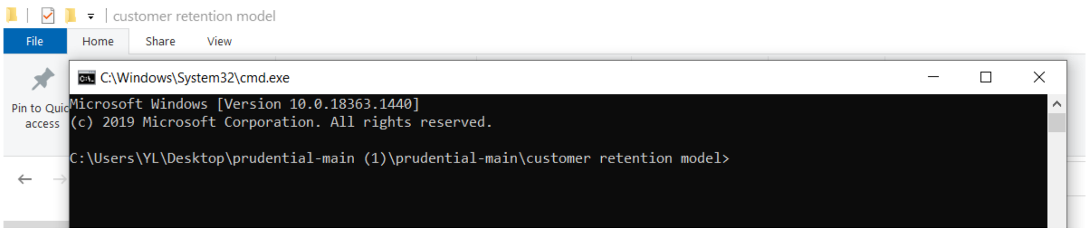
```
# Random Forest
clf_rf = RandomForestClassifier(n_estimators=400, min_samples_split=5, min_samples_leaf=1, max_features= 'sqrt', max_depth=None,bootstrap=
```



```
# Extra Trees
clf_et = ExtraTreesClassifier(criterion='gini', max_depth=32, max_features='sqrt',n_estimators= 50)
```

e. Save app.py using Ctl + S

4) Go back to your cmd

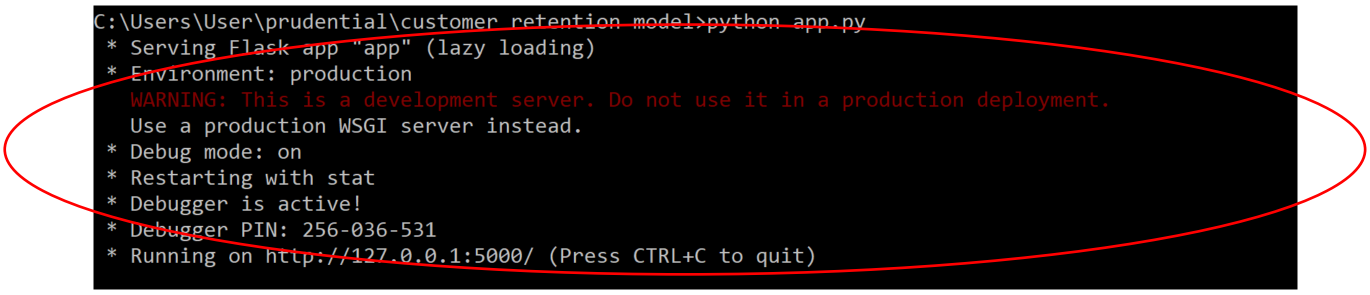


```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\YL\Desktop\prudential-main (1)\prudential-main\customer retention model>
```

5) Run the Flask App by typing:

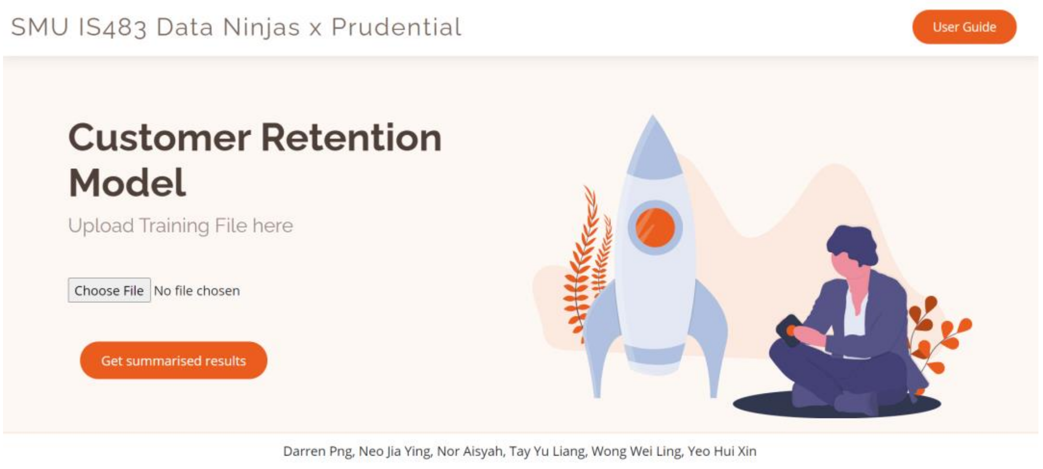
```
python app.py
```

You should see this which means the flask app is live and running.



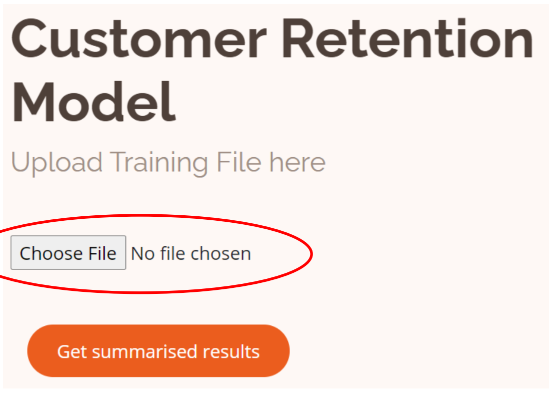
```
C:\Users\User\prudential\customer retention model>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 256-036-531
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

6) Go to the link: http://127.0.0.1:5000/ and you should see this page below:



SMU IS483 Data Ninjas x Prudential
User Guide
Customer Retention Model
Upload Training File here
Choose File
No file chosen
Get summarised results
Darren Png, Neo Jia Ying, Nor Aisyah, Tay Yu Liang, Wong Wei Ling, Yeo Hui Xin

7) Upload cleaned training file by clicking on the “Choose File” button.



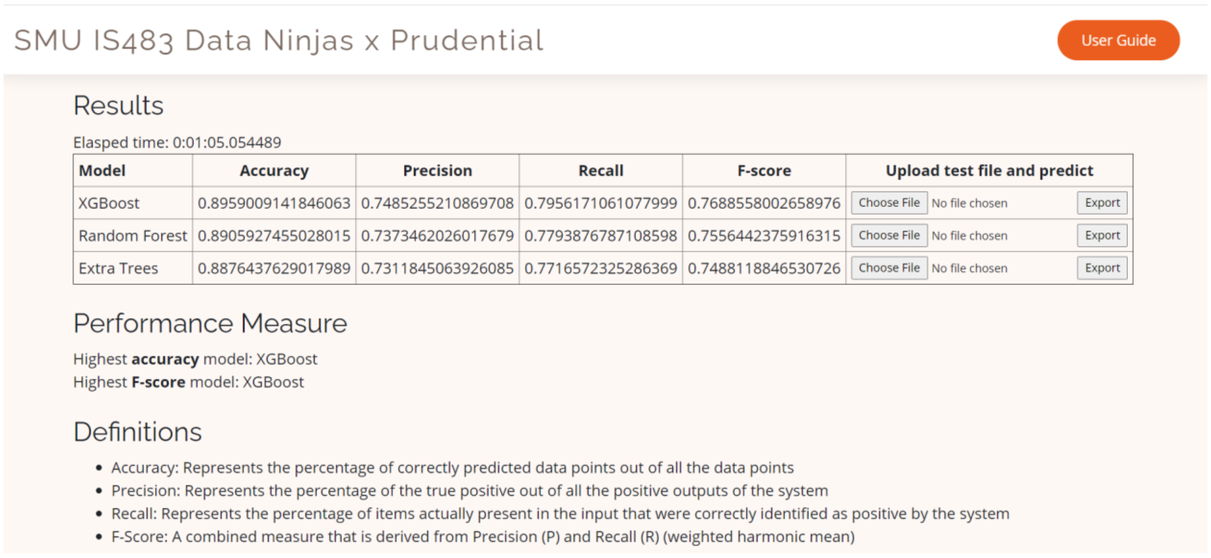
Customer Retention Model

Upload Training File here

**** Note: Training file needs to contain at least 10,000 rows of data**

8) Click on ‘Get summarised results’ button to get the metrics for these models: Random Forest, Extra Trees and XGBoost.

9) After clicking on ‘Get summarized results’, you will need to wait for a while, depending on how much data you have uploaded, before seeing this page. At this page, you can see the accuracy, precision, recall and f-score of the 3 models.



SMU IS483 Data Ninjas x Prudential

Results

Elasped time: 0:01:05.054489

Model	Accuracy	Precision	Recall	F-score	Upload test file and predict
XGBoost	0.8959009141846063	0.7485255210869708	0.7956171061077999	0.7688558002658976	Choose File No file chosen Export
Random Forest	0.8905927455028015	0.7373462026017679	0.7793876787108598	0.7556442375916315	Choose File No file chosen Export
Extra Trees	0.8876437629017989	0.7311845063926085	0.7716572325286369	0.7488118846530726	Choose File No file chosen Export

Performance Measure

Highest **accuracy** model: XGBoost
Highest **F-score** model: XGBoost

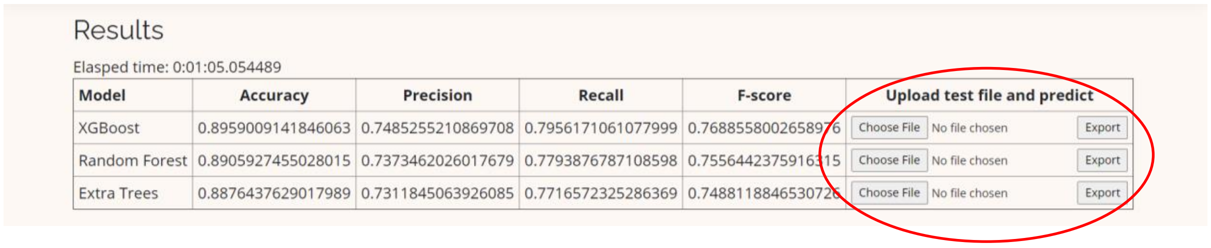
Definitions

- Accuracy: Represents the percentage of correctly predicted data points out of all the data points
- Precision: Represents the percentage of the true positive out of all the positive outputs of the system
- Recall: Represents the percentage of items actually present in the input that were correctly identified as positive by the system
- F-Score: A combined measure that is derived from Precision (P) and Recall (R) (weighted harmonic mean)

10) To choose the best model to predict the labels on your test file:

For the model chosen

i) Upload test file
ii) Click “export”



Results

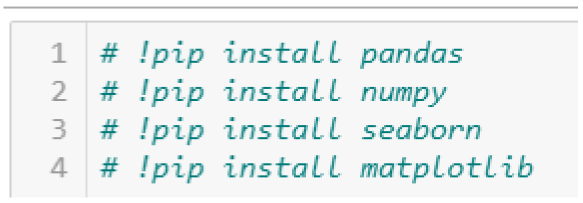
Elasped time: 0:01:05.054489

Model	Accuracy	Precision	Recall	F-score	Upload test file and predict
XGBoost	0.8959009141846063	0.7485255210869708	0.7956171061077999	0.7688558002658976	Choose File No file chosen Export
Random Forest	0.8905927455028015	0.7373462026017679	0.7793876787108598	0.7556442375916315	Choose File No file chosen Export
Extra Trees	0.8876437629017989	0.7311845063926085	0.7716572325286369	0.7488118846530726	Choose File No file chosen Export

11) Once you click on ‘export’, a file will be downloaded. This file contains the labels predicted by the respective model on the test file you uploaded.

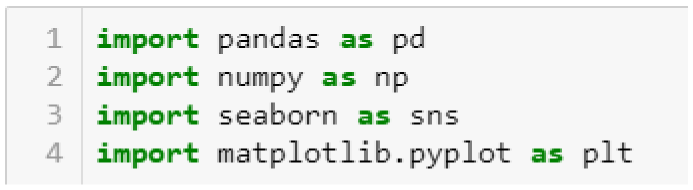
Section 4: Clustering Model

1) Open Jupyter notebook
2) Open “Clustering for Customer who churned.ipynb” OR “Clustering for Customer who retained.ipynb”
3) Uncomment and run the first cell to install pandas, numpy, seaborn and matplotlib.



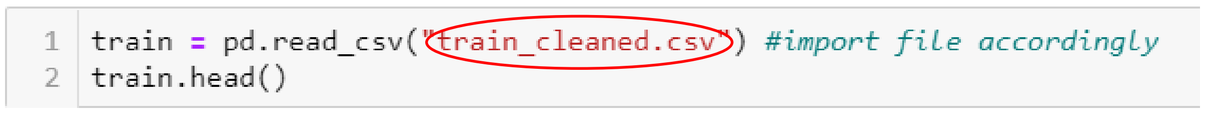
```
# !pip install pandas
# !pip install numpy
# !pip install seaborn
# !pip install matplotlib
```

4) Run the 2nd cell to import packages



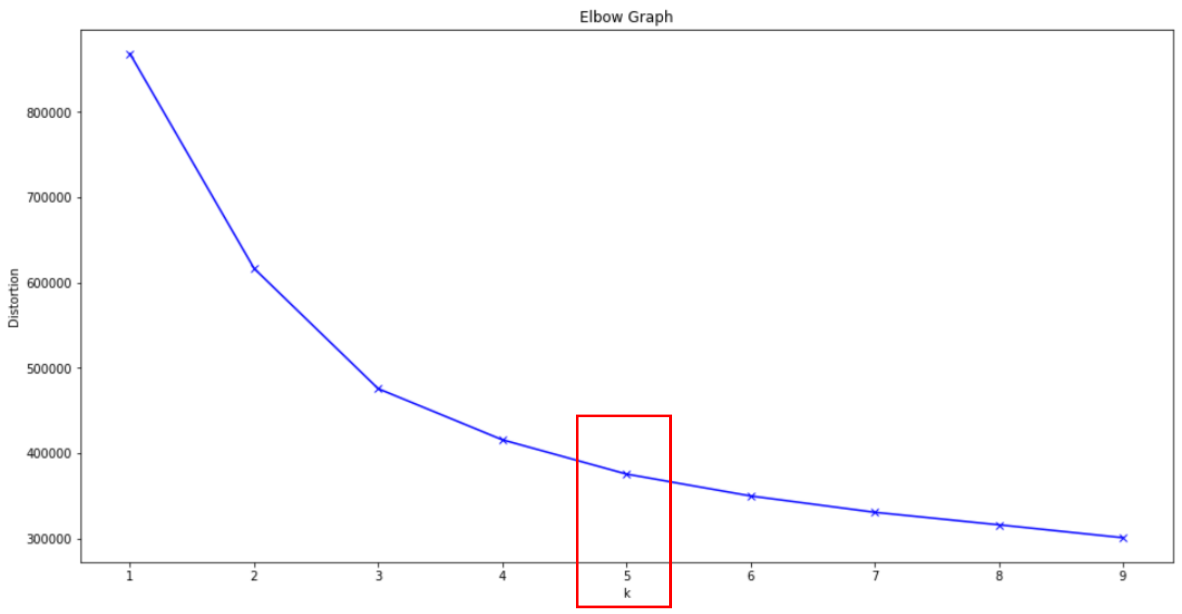
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

5) Import the test file that you exported in section 2 (the file with labels predicted). Change the file name to import accordingly.



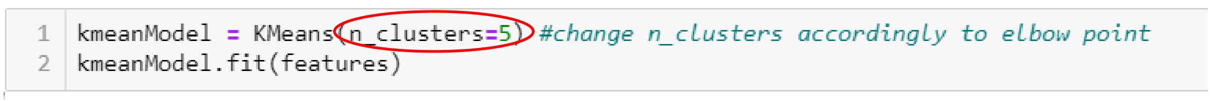
```
train = pd.read_csv("train_cleaned.csv") #import file accordingly
train.head()
```

6) Run all the cells until you reach the elbow plot.



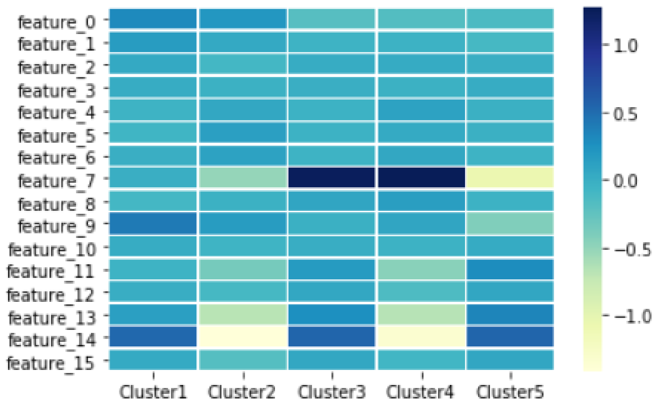
Elbow point is when the decrease in distortions is not as significant as k increases. In this case, the elbow point occurs when k=5.

7) For Kmeans clustering, change n_clusters to the elbow point’s k.



```
kmeanModel = KMeans(n_clusters=5) #change n_clusters accordingly to elbow point
kmeanModel.fit(features)
```

8) Continue running the cells until you reach this heatmap.



This heatmap tells you the highly ranked variables in each cluster. The highly ranked variables are shaded in very light yellow or very dark blue.