This is the second part of the Telcom Churn Analysis

In [1]:

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

In [2]:

```
from sklearn.preprocessing import StandardScaler,RobustScaler,QuantileTransformer
from sklearn.feature_selection import SelectKBest,mutual_info_classif
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [3]:

```
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
```

In [4]:

```
df = pd.read_csv("/content/drive/MyDrive/DSC 550/data.csv")
df.head()
```

Out[4]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	Churn	gender_Male	Partner_Yes	Dependents_Yes	PhoneService	NonMult_PhoneService	...	NoStr
0	0	1	29.85	29.85	0	0	1	0	0	1	...	
1	0	34	56.95	1889.50	0	1	0	0	1	0	...	
2	0	2	53.85	108.15	1	1	0	0	1	0	...	
3	0	45	42.30	1840.75	0	1	0	0	0	1	...	
4	0	2	70.70	151.65	1	0	0	0	1	0	...	

5 rows × 31 columns

In [5]:

```
df['Churn'].value_counts()
```

Out[5]:

```
0    5163
1    1869
Name: Churn, dtype: int64
```

In [6]:

```
X = df.drop('Churn',axis=1)
y = df['Churn']
```

In [7]:

```
undersample = RandomUnderSampler(sampling_strategy='majority')
X_over,y_over = undersample.fit_resample(X,y)
```

In [8]:

```
y_over.value_counts()
```

Out[8]:

```
0    1869
1    1869
Name: Churn, dtype: int64
```

In [9]:

```
from sklearn.model_selection import train_test_split
```

In [10]:

```
# Get X_train, X_test, y_train and y_test

X_train, X_test, y_train, y_test = train_test_split(X_over, y_over, test_size=0.20, random_state=42)
```

Setting up a pipeline

In [11]:

```
pipe = Pipeline([('scaler',StandardScaler()),
                 ('selector',SelectKBest(mutual_info_classif,k=7)),
                 ('classifier',LogisticRegression())])
```

Define the search Space

In [12]:

```
search_space = [{'classifier':[LogisticRegression()],
                 'classifier__penalty':['l1','l2'],
                 'classifier__C':np.logspace(0,4,10)},
                {'classifier':[KNeighborsClassifier()],
                'classifier__n_neighbors':[3,5,8,11],
                 'classifier__weights':['uniform','distance']},
                {'classifier':[RandomForestClassifier()],
                 'classifier__n_estimators':[10,100,1000],
                 'classifier__max_features':[1,2,3]}]
```

In [13]:

```
# Run the gridsearch

model = GridSearchCV(pipe,search_space,cv=10,verbose=0)
model.fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
100 fits failed out of a total of 370.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
100 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/pipeline.py", line 394, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py", line 449, in _check_solver
    % (solver, penalty)
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test scores are non-f
inite: [       nan 0.75183946        nan 0.74782609        nan 0.75150502
        nan 0.75317726        nan 0.75150502        nan 0.74782609
        nan 0.75351171        nan 0.75117057        nan 0.75217391
        nan 0.75083612 0.7173913  0.70468227 0.73344482 0.71471572
 0.74080268 0.71371237 0.75652174 0.71973244 0.70100334 0.72207358
 0.71505017 0.70267559 0.71103679 0.71371237 0.69598662 0.71003344
 0.71304348]
  category=UserWarning,
```

Out[13]:

```
GridSearchCV(cv=10,
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('selector',
                                        SelectKBest(k=7,
                                                    score_func=<function mutual_info_classif at 0x7f14623fd0e0>)),
                                       ('classifier', LogisticRegression())]),
             param_grid=[{'classifier': [LogisticRegression()],
                          'classifier__C': array([1.00000000e+00, 2.78255940e+00, 7.74263683e+00, 2.15443469e+01,
       5.99484250e+01, 1.668...64158883e+02, 1.29154967e+03,
       3.59381366e+03, 1.00000000e+04]),
                          'classifier__penalty': ['l1', 'l2']},
                         {'classifier': [KNeighborsClassifier(n_neighbors=11)],
                          'classifier__n_neighbors': [3, 5, 8, 11],
                          'classifier__weights': ['uniform', 'distance']},
                         {'classifier': [RandomForestClassifier()],
                          'classifier__max_features': [1, 2, 3],
                          'classifier__n_estimators': [10, 100, 1000]}])
```

Obtain predictions for train and test sets

In [14]:

```
best_estm = model.best_estimator_
print(best_estm)
```

```
Pipeline(steps=[('scaler', StandardScaler()),
                ('selector',
                 SelectKBest(k=7,
                             score_func=<function mutual_info_classif at 0x7f14623fd0e0>)),
                ('classifier', KNeighborsClassifier(n_neighbors=11))])
```

In [15]:

```
from sklearn.metrics import mean_squared_error
```

In [16]:

```
y_pred_train = best_estm.predict(X_train)
mse = mean_squared_error(y_pred_train,y_train)
print("MSE: %2f" %mse)
```

```
MSE: 0.219732
```

In [17]:

```
y_pred_test = best_estm.predict(X_test)
mse = mean_squared_error(y_pred_test,y_test)
print("MSE: %2f" %mse)
```

```
MSE: 0.260695
```

Print classification Report

In [18]:

```
from sklearn.metrics import confusion_matrix,classification_report
```

In [19]:

```
cm = confusion_matrix(y_test,y_pred_test)
```

In [20]:

```
print(cm)
```

```
[[269 110]
 [ 85 284]]
```

In [21]:

```
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.76      0.71      0.73       379
           1       0.72      0.77      0.74       369

    accuracy                           0.74       748
   macro avg       0.74      0.74      0.74       748
weighted avg       0.74      0.74      0.74       748
```

Having adjusted my class size using random undersampler, the classification of those that churn has improved from 55% to 75%. I also incorporated feature selection into the pipeline model to reduce unnecessary noise.

In [21]: