Brandeis University
International Business School

Machine Learning Project:
A case study on Telecom Company Data

Bus/Fin 241F

Team Members:
Jiawang Zhou,
Jiawei Zhang,
Luhan Shen,
Nan Cheng,
Rufeng Zhu,
Shengzhe Xu,
Yao Long

Part 1 Data Description

We use the data tells the churn rate of a certain Telcom company, containing 21 variables. The dependent variable is churn or not. And the detailed description of the data is listed below.

Table 1: Data Description

Variable Name	Description	Expectation
Customer ID	Identification of the customer in the company	--
Gender	Whether the customer is a male or a female	We do not expect differences between the churn rate of male and female.
SeniorCitizen	Whether the customer is a senior citizen or not	If the customer is a senior citizen, we expect the customer have a higher churn possibility because senior citizen are more likely to compare the prices between different Telcom.
Partner	Whether the customer has a partner or not	If the customer has a partner, we expect the customer has a lower churn possibility since the customer may sign the contract together with the partner.
Dependents	Whether the customer has a dependent or not	If the customer has a dependent, we expect the customer has a lower churn possibility since the customer may sign the contract together with the dependent.
Tenure	Number of months the customer has stayed with the company	We expect the churn rate to lower with the increase in the time of the tenure.
Phone Services	Whether the customer has phone services or not	If the customer has phone services. We expect the churn rate to be lower because the with the phone services it would be more difficult to change a Telcom.
MultipleLines	Whether the customer has multiple lines or not	If the customer has multiple lines, we expect the churn rate to be lower.
InternetServices	Customer's internet service provider (DSL, Fiber optic, No)	We expect the churn rate to be the lowest with Fiber optic Internet services since this iis the fastest option.
OnlineSecurity	Whether the customer has online security or not	If the customer has an online security, we expect the churn rate to be lower since with the security, the customer would be more willing to stay with the company.
OnlineBackup	Whether the customer has online backup or not	If the customer has an online backup, we expect the churn rate to be lower since with the backup, the customer would be more willing to stay with the company.

Table 1: Data Description (Continued)

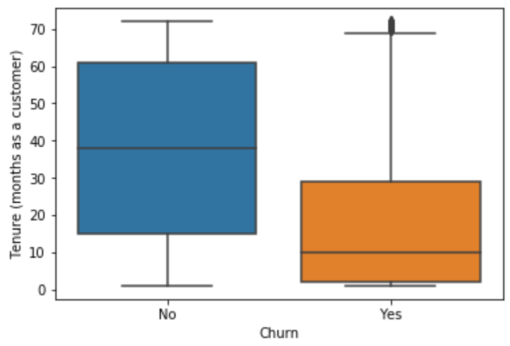
Variable Name	Description	Expectation
DeviceProtection	Whether the customer has a device protection or not	If the customer has a device protection, we expect the churn rate to be lower since with the protection, the customer would be more willing to stay with the company.
TechSupport	Whether the customer has technology support or not	If the customer has a technology support, we expect the churn rate to be lower since with the support, the customer would be more willing to stay with the company.
StreamingTV	Whether the customer has streaming TV or not	If the customer has a streaming TV, we expect the churn rate to be higher since we find out that this company's charges on TV and Movie are more expensive than its competitors.
StreamingMovies	Whether the customer has streaming Movies or not	If the customer has a streaming Movie, we expect the churn rate to be higher since we find out that this company's charges on TV and Movie are more expensive than its competitors.
Contract	The contract term of the customer (Month-to-month, One year, Two year)	We expect the churn rate to be lower with the increase in the duration of the contract.
PaperlessBilling	Whether the customer has paperless billing or not	We expect the churn rate to be lower if the customer has a paperless billing.
PaymentMethod	The customer's payment method (Electronic check, mailed check, Bank transfer (automatic), Credit card (automatic))	We expect the churn rate to be lower if the customer pays by automatic credit card.
MonthlyCharges	The amount charged to the customer monthly	We expect the churn rate to be higher with the increase in the monthly charges.
TotalCharges	The total amount charged to the customer	We expect the churn rate to be higher with the increase in the total charges.

Part 2 Data Visualization

We then did the data visualization to understand the data. We choose the variables: Tenure, Senior Citizen, Contract type and monthly charges.

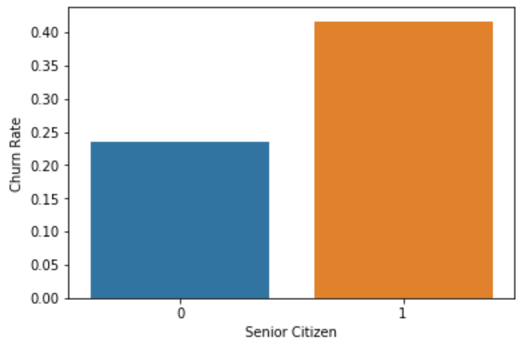
- According to the chart, customers are more likely to churn to other companies when the tenure is short and would not churn when the tenure duration is over 30 months.

Graph 1: Churn Choice VS Tenure



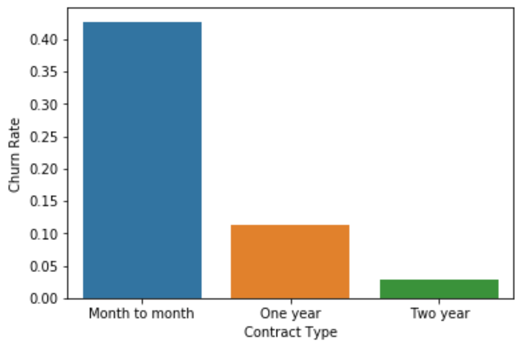
- According to the chart, we also found out that if the customer is a senior citizen, then the customer has a higher possibility to churn than non-senior citizen.

Graph 2: Seniority Vs Churn Rate



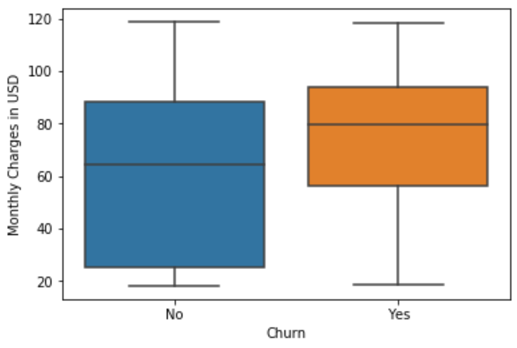
- The type of contract also has an obvious relationship with the churn rate. When the contract is signed up monthly, the customer is much more likely to churn than the customer who sign the contract yearly or every two years.

Graph 3: Contract Types Vs Churn Rate

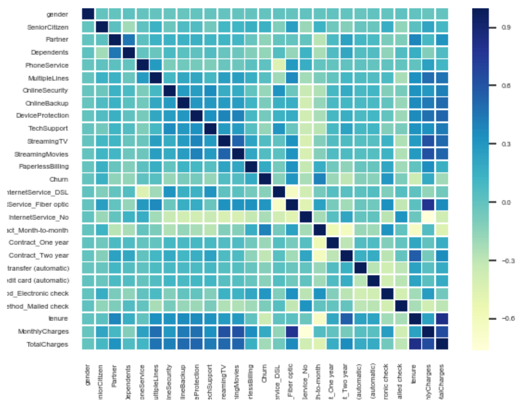


- According to the chart, when the monthly charge is higher than $50 per month, customer may start to churn. There is no customer churn under $50, while all the customer would churn to other companies if the monthly charge exceeds approximately $80 per month.

Graph 4: Churn Choice Vs Monthly Charge



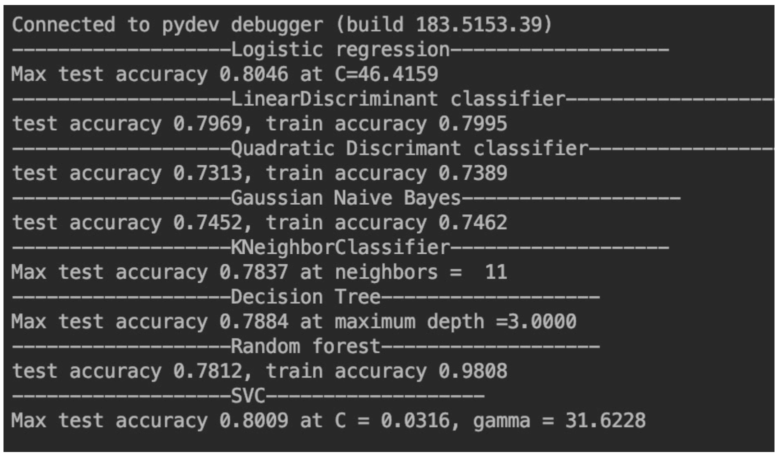
Graph 5: Heat Map



Part 3 Model Selection

In this project, we aim to find the best model for the real-world, Telecom customer churn rate dataset. Using the same metric, mean test score, to decide the best model, we want to know if the conclusion we achieved in Problem 3 Question 4, which used a synthesized dataset, also works for a real-world dataset. To maximize the accuracy of our conclusion, we conducted cross validation and collected the best accuracy for each model. The test accuracy for different models are as follows:

Graph 6: Monte Carlo Result



```
Connected to pydev debugger (build 183.5153.39)
-------------------Logistic regression-------------------
Max test accuracy 0.8046 at C=46.4159
-------------------LinearDiscriminant classifier------------------
test accuracy 0.7969, train accuracy 0.7995
-------------------Quadratic Discrimant classifier----------------
test accuracy 0.7313, train accuracy 0.7389
-------------------Gaussian Naive Bayes-------------------
test accuracy 0.7452, train accuracy 0.7462
-------------------KNeighborClassifier-------------------
Max test accuracy 0.7837 at neighbors =  11
-------------------Decision Tree-------------------
Max test accuracy 0.7884 at maximum depth =3.0000
-------------------Random forest-------------------
test accuracy 0.7812, train accuracy 0.9808
-------------------SVC-------------------
Max test accuracy 0.8009 at C = 0.0316, gamma = 31.6228
```

- From the result above, we can find that the best model is Logistic regression, followed by SVC which has a very close test accuracy. This is very different from what we got in Problem Set 3, where SVC performed the worst. It's excited to find that logistic regression gives the best model, because we can achieve good interpretation without sacrificing for accuracy.
- KNN classifier performed pretty well when we made a loop and tested different k from 3 to 13 and found that k=11 gives the best test accuracy of 0.7837, which is better than the Random Forest model.
- Decision tree was the best model in Problem Set 3, but here it performs just almost as well as the Linear Discriminant. We've imagined that Linear Discriminant would do better when restricted to real values, and our result shows that it does.
- In this run, the Random Forest model didn't do better compared to the maximum test accuracy of Decision Tree, when we have tested different depth of trees from 1 to 20. But the result of Random Forest could change due to the randomness.
- Quadratic discriminant is even worse than the Gaussian Naïve Bayes, therefore, it should not be considered a good model.

Part 4 Appendix of Python Codes

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns#visualization
import warnings

warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import
confusion_matrix,accuracy_score,classification_report
from sklearn.metrics import roc_auc_score,roc_curve,scorer
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score,recall_score
from sklearn.tree import export_graphviz
#from graphviz import Source
from IPython.display import SVG,display



#1. Data manipulation
telcom=pd.read_csv("/Users/zhoujiawang/Desktop/WA_Fn-UseC_-Telco-Customer-
Churn.csv")
telcom.head()
telcom.nunique()
telcom.dtypes

#Replacing spaces with null values in total charges column
telcom['TotalCharges'] = telcom["TotalCharges"].replace(" ",np.nan)
#Dropping null values from total charges column which contain 15% missing
data
telcom = telcom[telcom["TotalCharges"].notnull()]
telcom = telcom.reset_index()[telcom.columns]
#convert to float type
telcom["TotalCharges"] = telcom["TotalCharges"].astype(float)

#replace 'No internet service' to No for the following columns
replace_cols = [ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
                'TechSupport','StreamingTV', 'StreamingMovies']
for i in replace_cols :
    telcom[i]= telcom[i].replace({'No internet service' : 'No'})
#replace 'No pgone service' to No for the following columns
telcom['MultipleLines']  = telcom['MultipleLines'].replace({'No phone
service' : 'No'})
#replace values
telcom["SeniorCitizen"] = telcom["SeniorCitizen"].replace({1:"Yes",0:"No"})
telcom=telcom.drop(columns='customerID',axis = 1)

#2.visulization
plt.interactive(False)
figure_nbr=1
figure_nbr+=1
plt.figure(figure_nbr)
sns.boxplot(y="tenure",x="Churn", data=telcom)
plt.ylabel("Tenure (months as a customer)")
plt.xlabel("Churn")
plt.show()

figure_nbr+=1
plt.figure(figure_nbr)
sns.barplot(x=[0,1],y=[0.236,0.417])
plt.ylabel("Churn Rate")
plt.xlabel("Senior Citizen")
plt.show()

figure_nbr+=1
plt.figure(figure_nbr)
sns.barplot(x=["Month to month","One year","Two
year"],y=[0.427,0.113,0.028])
plt.ylabel("Churn Rate")
plt.xlabel("Contract Type")
plt.show()

figure_nbr+=1
plt.figure(figure_nbr)
sns.boxplot(y="MonthlyCharges",x="Churn", data=telcom)
```

```
plt.ylabel("Monthly Charges in USD")
plt.xlabel("Churn")
plt.show()

#3.Data pre-processing
#categorical columns
cat_cols   = telcom.nunique()[telcom.nunique() < 6].keys().tolist()

target_col = ['Churn']
cat_cols   = [x for x in cat_cols if x not in target_col]

#numerical columns
num_cols   = [x for x in telcom.columns if x not in cat_cols + target_col]
#Binary columns with 2 values
bin_cols   = telcom.nunique()[telcom.nunique() == 2].keys().tolist()
#Columns more than 2 values
multi_cols = [i for i in cat_cols if i not in bin_cols]

#Label encoding Binary columns
le = LabelEncoder()
for i in bin_cols :
    telcom[i] = le.fit_transform(telcom[i])

#create dummy for multi-value columns
telcom = pd.get_dummies(data = telcom,columns = multi_cols )

#Scaling Numerical columns
std = StandardScaler()
scaled = std.fit_transform(telcom[num_cols])
scaled = pd.DataFrame(scaled,columns=num_cols)

#dropping original values merging scaled values for numerical columns
df_telcom_og = telcom.copy()
telcom = telcom.drop(columns = num_cols,axis = 1)
telcom = telcom.merge(scaled,left_index=True,right_index=True,how = "left")


correlation = telcom.corr()

figure_nbr=1
plt.figure(figure_nbr)
sns.set(font_scale=.5)
sns.heatmap(correlation,xticklabels=correlation.columns,yticklabels=correlat
ion.columns,cmap="YlGnBu",linewidths=.5)
```

```
plt.show()
sns.set(font_scale=1)

#4. Modeling

##seperating dependent and independent variables
features = [i for i in telcom.columns if i not in target_col]
X = telcom[features]
y = telcom[target_col]

#4.1 logistic
#find the best tune: C
print('-------------------Logistic regression-------------------')
C = np.power(10, np.linspace(-5,5,10))
nmc=100
Test = np.zeros(nmc)
Train = np.zeros(nmc)
trainScore = np.zeros(len(C))
testScore = np.zeros(len(C))
for c in range(len(C)):
    lr = LogisticRegression(penalty='l2', C=C[c], solver='liblinear')
    for i in  range(nmc):
        X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size=0.25)
        trainFit=lr.fit(X_train,y_train)
        Train[i] =  trainFit.score(X_train,y_train)
        Test[i] =  trainFit.score(X_test,y_test)
    trainScore[c] = np.mean(Train)
    testScore[c] = np.mean(Test)
print('Max test accuracy {0:4.4f} at 
C={1:4.4f}'.format(np.max(testScore),C[np.argmax(testScore)]))

figure_nbr+=1
plt.figure(figure_nbr,figsize=(12,8))
plt.semilogx(C,testScore,label='test accuracy')
plt.semilogx(C,trainScore,label='train accuracy')
plt.legend()
plt.ylabel("Accuracy")
plt.xlabel("C")
plt.title('Logistic: training/test data accuracy over different C')
plt.show()

#4.2 LinearDiscriminant classifier
print('-------------------LinearDiscriminant classifier-------------------')
```

```
lda = LinearDiscriminantAnalysis()
trainScore=[]
testScore=[]
for i in range(nmc):
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
   trainFit=lda.fit(X_train,y_train)
   trainScore.append(trainFit.score(X_train,y_train))
   testScore.append(trainFit.score(X_test,y_test))
print('test accuracy {0:4.4f}, train accuracy
{1:4.4f}'.format(np.mean(testScore),np.mean(trainScore)))



#4.3 Quadratic Discrimant classifier
print('-------------------Quadratic Discrimant classifier-------------------
')
qda =  QuadraticDiscriminantAnalysis()
trainScore = []
testScore = []
for i in range(nmc):
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
   trainFit = qda.fit(X_train, y_train)
   trainScore.append(trainFit.score(X_train, y_train))
   testScore.append(trainFit.score(X_test, y_test))
print('test accuracy {0:4.4f}, train accuracy
{1:4.4f}'.format(np.mean(testScore),np.mean(trainScore)))

#4.4 Naive Bayes
print('-------------------Gaussian Naive Bayes-------------------')
gnb = GaussianNB()
trainScore = []
testScore = []
for i in range(nmc):
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
   trainFit = gnb.fit(X_train, y_train)
   trainScore.append(trainFit.score(X_train, y_train))
   testScore.append(trainFit.score(X_test, y_test))
print('test accuracy {0:4.4f}, train accuracy
{1:4.4f}'.format(np.mean(testScore),np.mean(trainScore)))

#4.5 knn
print('-------------------KNeighborClassifier-------------------')
K = np.arange(3,13,2)
trainScore = np.zeros(len(K))
```

```
testScore = np.zeros(len(K))
Test = np.zeros(nmc)
Train = np.zeros(nmc)
for k in range(len(K)):
    knn = KNeighborsClassifier(n_neighbors=K[k])
    for i in range(nmc):
        X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25)
        trainFit=knn.fit(X_train,y_train)
        Train[i] = trainFit.score(X_train, y_train)
        Test[i] = trainFit.score(X_test, y_test)
    trainScore[k] = np.mean(Train)
    testScore[k] = np.mean(Test)

print('Max test accuracy {0:4.4f} at neighbors
={1:4.0f}'.format(np.max(testScore),K[np.argmax(testScore)]))

figure_nbr+=1
plt.figure(figure_nbr,figsize=(12,8))
plt.plot(K,testScore,label='test accuracy')
plt.plot(K,trainScore,label='train accuracy')
plt.legend()
plt.ylabel("Accuracy")
plt.xlabel("Neighbors")
plt.title('KNN: training/test data accuracy over different neighbors')
plt.show()


#4.6 decision tree
print('--------------------Decision Tree--------------------')
max_depth = np.arange(1,10)
trainScore = np.zeros(len(max_depth))
testScore = np.zeros(len(max_depth))
Test = np.zeros(nmc)
Train = np.zeros(nmc)

for d in range(len(max_depth)):
    tree = DecisionTreeClassifier(max_depth=max_depth[d])
    for i in range(nmc):
        X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25)
        trainFit = tree.fit(X_train, y_train)
        Train[i] = trainFit.score(X_train, y_train)
        Test[i] = trainFit.score(X_test, y_test)
    trainScore[d] = np.mean(Train)
    testScore[d] = np.mean(Test)

print('Max test accuracy {0:4.4f} at maximum depth
={1:4.4f}'.format(np.max(testScore),max_depth[np.argmax(testScore)]))

figure_nbr+=1
plt.figure(figure_nbr,figsize=(12,8))
plt.plot(max_depth,testScore,label='test accuracy')
plt.plot(max_depth,trainScore,label='train accuracy')
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Max depth")
plt.title('Decision tree: training/test data accuracy over different Max
depth')
plt.show()


#trainFit=DecisionTreeClassifier(max_depth=max_depth[np.argmax(testScore)]).
fit(X_train,y_train)
'''graph = Source(export_graphviz(trainFit,out_file=None,
                                        rounded=True,proportion = False,
                                        feature_names = X_train.columns,
                                        precision  = 2,
                                        class_names=["Not churn","Churn"],
                                        filled = True
                                       ))
display(graph)'''



#4.7 random forest
#can try different parameters
#forest =
RandomForestClassifier(n_estimators=250,max_features=3,max_depth=5)
print('-------------------Random forest-------------------')
forest = RandomForestClassifier()
trainScore = []
testScore = []
for i in range(nmc):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
    trainFit = forest.fit(X_train, y_train)
    trainScore.append(trainFit.score(X_train, y_train))
    testScore.append(trainFit.score(X_test, y_test))
print('test accuracy {0:4.4f}, train accuracy
{1:4.4f}'.format(np.mean(testScore),np.mean(trainScore)))


#4.8 SVC
print('--------------------SVC--------------------')
C = np.power(10,np.linspace(-3,3,5))
gamma = np.power(10,np.linspace(-3,3,5))
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit
trainScore = np.zeros((len(C),len(gamma)))
testScore = np.zeros((len(C),len(gamma)))
Test = np.zeros(nmc)
Train = np.zeros(nmc)
for c in range(len(C)):
    for g in range(len(gamma)):
        svc = SVC(C=C[c], gamma=gamma[g], kernel='rbf')
        cvf = ShuffleSplit(n_splits=25, test_size=0.25)
        scores = cross_val_score(svc, X, y, cv=cvf)
        testScore[c,g] = np.mean(scores)

print('Max test accuracy {0:4.4f} at C = {1:4.4f}, gamma =
{2:4.4f}'.format(np.max(testScore), C[np.argmax(testScore) // 12],
gamma[np.argmax(testScore) % 12]))
```