Lab-Phase 7 Logistic Regression CV Telco_Churn

November 2, 2020

1 Logistic Regression w/Cross Validation

1.0.1 Import required packages

```
In [28]: %matplotlib inline

         from pathlib import Path

         import pandas as pd
         from sklearn import preprocessing
         import missingno as msno

         from sklearn.decomposition import PCA

         from sklearn.model_selection import train_test_split, GridSearchCV

         from sklearn import metrics
         from sklearn.metrics import accuracy_score

         from sklearn.linear_model import LogisticRegression, LogisticRegressionCV

         import seaborn as sns
         import matplotlib.pylab as plt

         from dmba import classificationSummary, gainsChart, liftChart
```

The Telco_Churn dataset was developed by IBM to model telecommunications customer relations in business analytics. Our goal is to predict the likelihood of a customer leaving the firm. This is an important issue, notably because attracting new customers have a high cost in both marketing promotions and lost revenue.

1.0.2 Load the data and perform initial inspection

```
In [29]: telco_df = pd.read_excel('../resource/lib/public/Telco_Churn.xlsx') # this is an Exce

         pd.set_option('display.max_rows', 50)
         pd.set_option('display.max_columns', 50)
```

```
In [30]: telco_df.head() # reproduce the output
```

```
Out[30]:    customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
         0  7590-VHVEG  Female              0     Yes         No       1           No
         1  5575-GNVDE    Male              0      No         No      34          Yes
         2  3668-QPYBK    Male              0      No         No       2          Yes
         3  7795-CFOCW    Male              0      No         No      45           No
         4  9237-HQITU  Female              0      No         No       2          Yes

               MultipleLines InternetService OnlineSecurity OnlineBackup  \
         0  No phone service             DSL             No          Yes
         1                No             DSL            Yes           No
         2                No             DSL            Yes          Yes
         3  No phone service             DSL            Yes           No
         4                No     Fiber optic             No           No

           DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
         0               No          No          No              No  Month-to-month
         1              Yes          No          No              No        One year
         2               No          No          No              No  Month-to-month
         3              Yes         Yes          No              No        One year
         4               No          No          No              No  Month-to-month

           PaperlessBilling              PaymentMethod  MonthlyCharges  TotalCharges  \
         0              Yes           Electronic check           29.85         29.85
         1               No               Mailed check           56.95       1889.50
         2              Yes               Mailed check           53.85        108.15
         3               No  Bank transfer (automatic)           42.30       1840.75
         4              Yes           Electronic check           70.70        151.65

           Churn
         0    No
         1    No
         2   Yes
         3    No
         4   Yes
```

```
In [31]: telco_df.shape # reproduce the output
```

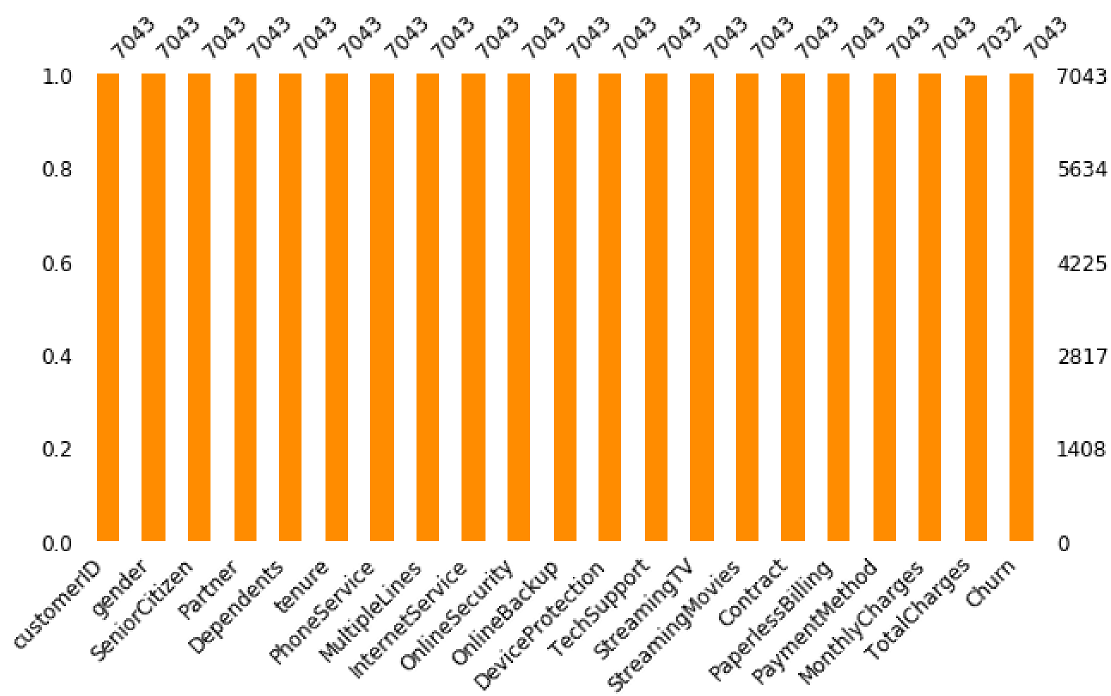
```
Out[31]: (7043, 21)
```

```
In [32]: telco_df.info() # reproduce the output
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
```

```
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7032 non-null   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

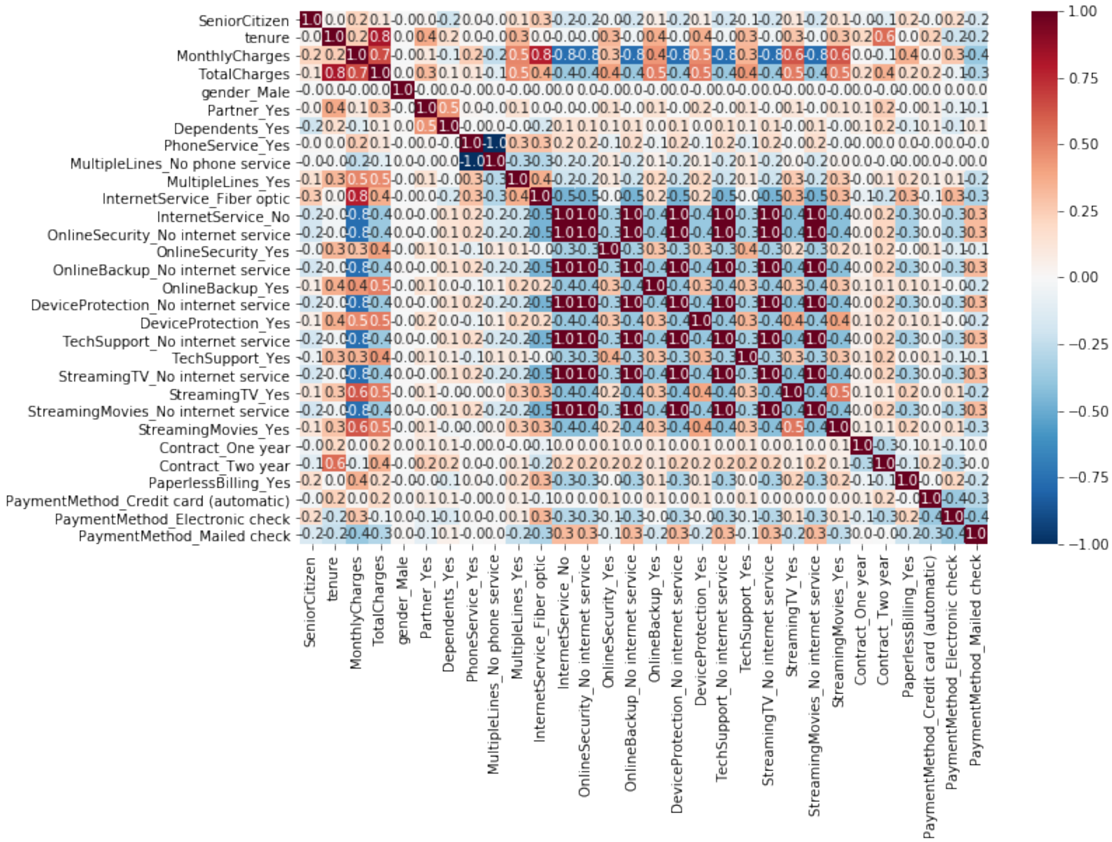
In [33]:
```python
# reproduce the output
%matplotlib inline
import missingno as msno
msno.bar(telco_df, color = "darkorange",figsize=(10,5), fontsize=12);
```



```
In [34]: telco_df = telco_df.drop('customerID', axis=1)

In [35]: # According to the missing value bar chart, only 11 samples are missing from a single

         telco_df = telco_df.dropna() # drop na values

In [36]: # Create a y response variable and an X collection of predictors. Remember, the outco

         y = telco_df['Churn']

         X = telco_df.drop(columns=['Churn'])

         print(len(X.columns))

19

In [37]: # Dummy code the set of predictors in preparation of logistic regression

         X = pd.get_dummies(X, prefix_sep="_", drop_first=True) # dummy code predictors

         print(len(X.columns))

30

In [38]: # Convert the text of Gone to a binary numeric variable (0/1)
         y = y.astype('category').cat.codes

         # Check for a class imbalance
         y.value_counts() # check value count

Out[38]: 0    5163
         1    1869
         dtype: int64

In [39]: print(pd.DataFrame(X.columns))

                                          0
0                             SeniorCitizen
1                                    tenure
2                            MonthlyCharges
3                              TotalCharges
4                               gender_Male
5                               Partner_Yes
6                            Dependents_Yes
```

```
7                         PhoneService_Yes
8           MultipleLines_No phone service
9                        MultipleLines_Yes
10             InternetService_Fiber optic
11                      InternetService_No
12       OnlineSecurity_No internet service
13                      OnlineSecurity_Yes
14         OnlineBackup_No internet service
15                        OnlineBackup_Yes
16     DeviceProtection_No internet service
17                    DeviceProtection_Yes
18          TechSupport_No internet service
19                         TechSupport_Yes
20          StreamingTV_No internet service
21                         StreamingTV_Yes
22      StreamingMovies_No internet service
23                     StreamingMovies_Yes
24                       Contract_One year
25                       Contract_Two year
26                    PaperlessBilling_Yes
27   PaymentMethod_Credit card (automatic)
28          PaymentMethod_Electronic check
29              PaymentMethod_Mailed check
```

In [40]:
```
# color-coded heatmap with correlation values

telco_df_corr = X.corr()

fig, ax = plt.subplots()

fig.set_size_inches(11, 7)

sns.heatmap(telco_df_corr, annot=True, fmt=".1f", cmap="RdBu_r", center=0, ax=ax)

plt.show()
```



1.0.3 Cross Validated Explanatory Model using GridSearchCV

```
In [41]: param_grid = {
             'penalty': ['l2'],
             'C' : [1e42],
             'solver': ['liblinear'],
             'class_weight': ['balanced'],
         }

In [42]: gridSearch = GridSearchCV(LogisticRegression(), param_grid, cv=10)

         gridSearch.fit(X, y)

         print('Initial parameters: ', gridSearch.best_params_)

         logit_reg = gridSearch.best_estimator_

         print('intercept ', logit_reg.intercept_[0])
         print(pd.DataFrame({'coeff': logit_reg.coef_[0]}, index=X.columns))
```

```
Initial parameters:  {'C': 1e+42, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'libl
intercept  0.1831070750196643
```

```
                                               coeff
SeniorCitizen                               0.225202
tenure                                     -0.055297
MonthlyCharges                              0.014624
TotalCharges                                0.000293
gender_Male                                -0.016834
Partner_Yes                                -0.006196
Dependents_Yes                             -0.156011
PhoneService_Yes                           -0.345413
MultipleLines_No phone service              0.528520
MultipleLines_Yes                           0.143280
InternetService_Fiber optic                 0.350194
InternetService_No                         -0.060817
OnlineSecurity_No internet service         -0.060817
OnlineSecurity_Yes                         -0.474165
OnlineBackup_No internet service           -0.060817
OnlineBackup_Yes                           -0.232989
DeviceProtection_No internet service       -0.060817
DeviceProtection_Yes                       -0.149205
TechSupport_No internet service            -0.060817
TechSupport_Yes                            -0.434944
StreamingTV_No internet service            -0.060817
StreamingTV_Yes                             0.052564
StreamingMovies_No internet service        -0.060817
StreamingMovies_Yes                         0.062734
Contract_One year                          -0.668991
Contract_Two year                          -1.368261
PaperlessBilling_Yes                        0.322900
PaymentMethod_Credit card (automatic)      -0.051417
PaymentMethod_Electronic check              0.335348
PaymentMethod_Mailed check                 -0.054520
```

Train/test split with stratification of the response variable

In [43]:
```
# Split the data into training and test sets (holdout approach)

train_X, test_X, train_y, test_y =  train_test_split(X,y, test_size= 0.5, stratify = y
```

In [44]:
```
train_X.shape
```

Out[44]:
```
(3516, 30)
```

In [45]:
```
train_y.value_counts()# value counts
```

Out[45]:
```
0    2582
1     934
dtype: int64
```

Fix the class imbalance issue on the training data (Skip this and return to check differences)

```
In [46]: from imblearn.over_sampling import ADASYN

         ada = ADASYN()

         train_X, train_y = ada.fit_sample(train_X, train_y.ravel())

         train_X = pd.DataFrame(train_X)
         train_y = pd.Series(train_y)
```

```
In [47]: # Check the synthetic insertions
         train_y.value_counts()# value counts
```

```
Out[47]: 1    2647
         0    2582
         dtype: int64
```

Feature scaling to prepare data for l1 regularization

```
In [48]: # Feature Scaling
         from sklearn.preprocessing import StandardScaler

         sc = StandardScaler()

         train_X_std = sc.fit_transform(train_X)
         test_X_std = sc.transform(test_X)
```

1.0.4 Predictive Model

```
In [49]: # Build a logistic regression model as a baseline

         logit_reg = LogisticRegressionCV(penalty="l1", Cs=100, solver='liblinear', class_weig
                                          cv=10, max_iter=5000, scoring="accuracy", random_sta

         model = logit_reg.fit(train_X, train_y)
```

```
In [50]: # we are only interested in classification accuracy
         classificationSummary(train_y, model.predict(train_X))# on train data
         classificationSummary(test_y, model.predict(test_X)) # on test data
```

```
Confusion Matrix (Accuracy 0.8281)

       Prediction
Actual    0    1
     0 2101  481
     1  418 2229
Confusion Matrix (Accuracy 0.7673)
```

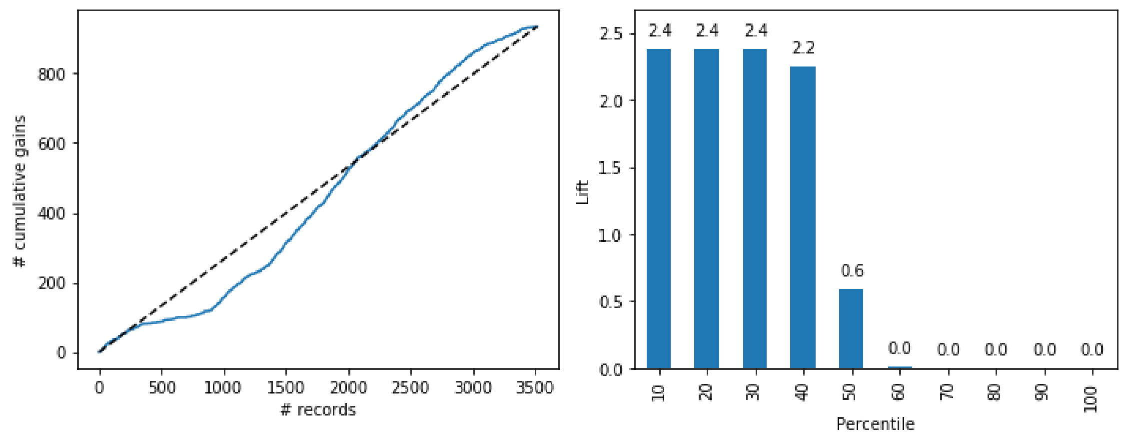
```
       Prediction
Actual    0    1
     0 2094  487
     1  331  604
```

```
In [51]: classes = model.predict(test_X_std)

         print(metrics.classification_report(test_y, classes))
```

```
              precision    recall  f1-score   support

           0       0.69      0.54      0.61      2581
           1       0.20      0.32      0.25       935

    accuracy                           0.49      3516
   macro avg       0.45      0.43      0.43      3516
weighted avg       0.56      0.49      0.51      3516
```

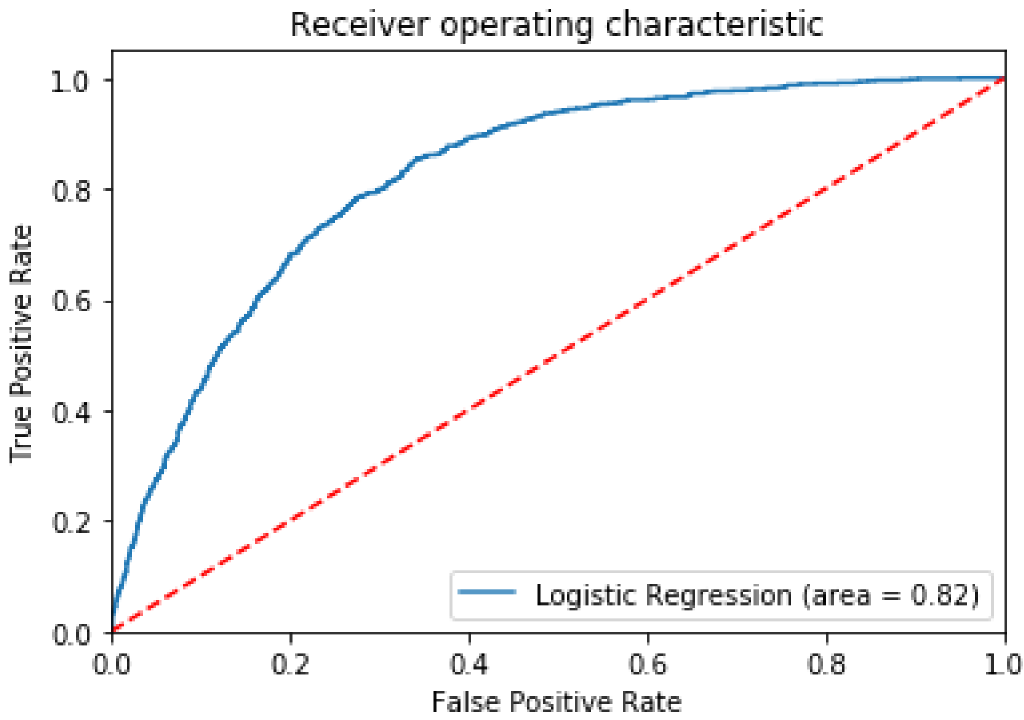
1.0.5 Lift and Gain Charts

```
In [52]: %matplotlib inline

         logit_reg_pred = logit_reg.predict(test_X_std)
         logit_reg_proba = logit_reg.predict_proba(test_X_std)
         logit_result = pd.DataFrame({'actual': test_y,
                                      'p(0)': [p[0] for p in logit_reg_proba],
                                      'p(1)': [p[1] for p in logit_reg_proba],
                                      'predicted': logit_reg_pred })

         df = logit_result.sort_values(by=['p(1)'], ascending=False)
         fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

         gainsChart(df.actual, ax=axes[0])
         liftChart(df['p(1)'], title=False, ax=axes[1])

         plt.tight_layout()
         plt.show()
```



1.0.6 ROC Chart

In [53]:
```
logit_reg_pred = logit_reg.predict(test_X)
logit_reg_proba = logit_reg.predict_proba(test_X)

preds = logit_reg_proba[:,1]
fpr, tpr, threshold = metrics.roc_curve(test_y, preds)
roc_auc = metrics.auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```


Receiver operating characteristic
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
Logistic Regression (area = 0.82)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

In [ ]:

In [ ]: