Using the data in the link below, attempt to model a customer's propensity to join our loyalty program

Ayan Karim

In [1]:
```
# Import Dependencies
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier, GradientBoostingCla
ssifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import confusion_matrix

from scipy.stats import boxcox
```

In [2]:
```
df = pd.read_csv('customers_data.csv')
```

Description of Data:

This data set contains demographic information on customers and labels whether they've joined the loyalty program. More specifically, it includes a mixture of categorical (binary), continuous and discrete variables on gender, age, whether there's a card on file, the purchase amount and the time elapsed since last purchase for each customer.

The original data set contained 12,000 rows of data and 7 columns, however after cleaning the data and under sampling to balance our classes, we have 9,741 rows of data and 6 columns that we actually use for modeling.

Explore Data

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	purch_amt	gender	card_on_file	age	days_since_last_purch	loyalty
0	0	19.58	male	no	31.0	35.0	False
1	1	65.16	male	yes	23.0	61.0	False
2	2	40.60	female	no	36.0	49.0	False
3	3	38.01	male	yes	47.0	57.0	False
4	4	22.32	female	yes	5.0	39.0	False

In [4]:
```
df.shape
```

Out[4]:
```
(120000, 7)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120000 entries, 0 to 119999
Data columns (total 7 columns):
Unnamed: 0               120000 non-null int64
purch_amt                120000 non-null float64
gender                   120000 non-null object
card_on_file             120000 non-null object
age                      120000 non-null float64
days_since_last_purch    120000 non-null float64
loyalty                  120000 non-null bool
dtypes: bool(1), float64(3), int64(1), object(2)
memory usage: 5.6+ MB
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'purch_amt', 'gender', 'card_on_file', 'age',
       'days_since_last_purch', 'loyalty'],
      dtype='object')
```

In [7]:
```
# Drop Unnamed column
df = df.drop(['Unnamed: 0'], axis=1)
```

In [8]:
```
df.describe()
```

Out[8]:

	purch_amt	age	days_since_last_purch
count	120000.000000	120000.000000	120000.000000
mean	44.036234	25.803008	56.605908
std	20.473148	10.153072	16.422187
min	-43.950000	-22.000000	-9.000000
25%	30.210000	19.000000	45.000000
50%	43.970000	26.000000	57.000000
75%	57.830000	33.000000	68.000000
max	142.200000	71.000000	125.000000

Clean Data

In [9]:
```
# Remove unrealistice values for continuous variables
df.purch_amt = df.purch_amt[df.purch_amt > df.purch_amt.quantile(.25)]
df.age = df.age[df.age > df.age.quantile(.25)]
df.days_since_last_purch = df.days_since_last_purch[df.days_since_last_p
urch > df.days_since_last_purch.quantile(.25)]

# Remove rows with null values
df = df.dropna()
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 47455 entries, 1 to 119999
Data columns (total 6 columns):
purch_amt                47455 non-null float64
gender                   47455 non-null object
card_on_file             47455 non-null object
age                      47455 non-null float64
days_since_last_purch    47455 non-null float64
loyalty                  47455 non-null bool
dtypes: bool(1), float64(3), object(2)
memory usage: 2.2+ MB
```

In [11]:
```
df.shape
```

Out[11]:
```
(47455, 6)
```

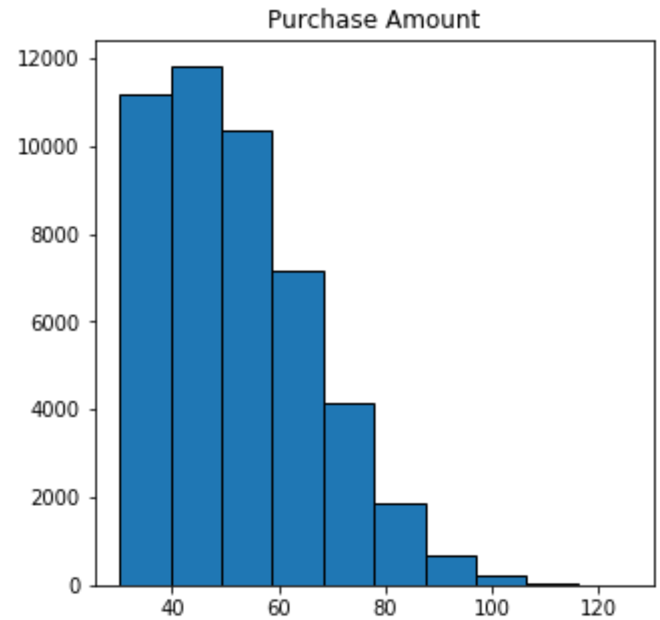
In [12]:
```
df.describe()
```

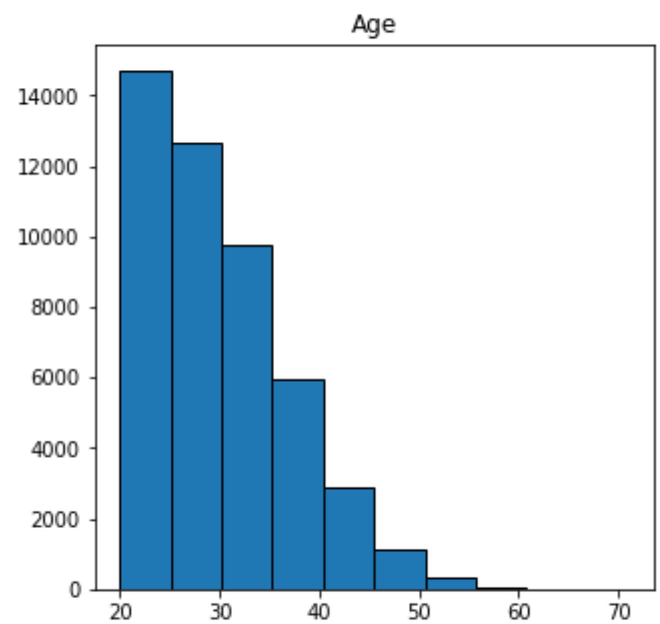
Out[12]:

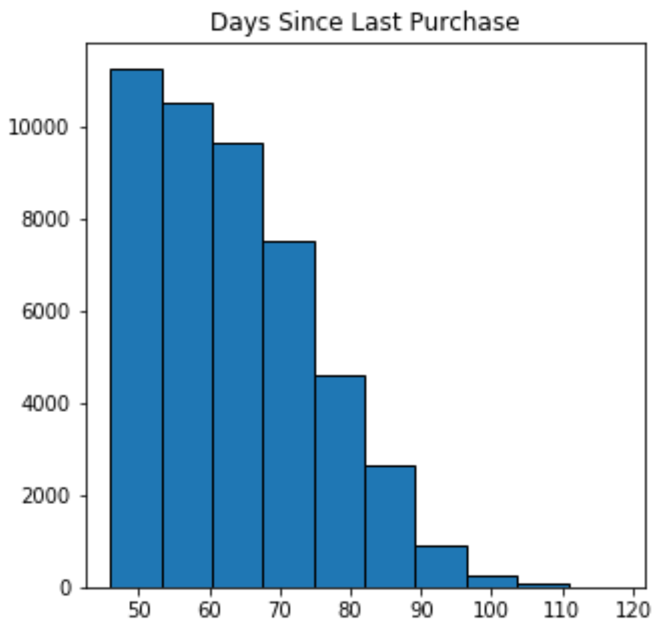
	purch_amt	age	days_since_last_purch
count	47455.000000	47455.000000	47455.00000
mean	52.051808	30.093373	63.38940
std	14.710210	7.180507	11.81037
min	30.220000	20.000000	46.00000
25%	40.300000	24.000000	54.00000
50%	49.870000	29.000000	62.00000
75%	61.360000	35.000000	71.00000
max	125.530000	71.000000	118.00000

Visualize Data

In [13]:
```
# Visualize distribution of continuous variables
plt.figure(figsize=(5,5))
plt.hist(df.purch_amt, edgecolor = 'k')
plt.title('Purchase Amount')
plt.show()

plt.figure(figsize=(5,5))
plt.hist(df.age, edgecolor = 'k')
plt.title('Age')
plt.show()

plt.figure(figsize=(5,5))
plt.hist(df.days_since_last_purch, edgecolor = 'k')
plt.title('Days Since Last Purchase')
plt.show()
```


Purchase Amount
12000
10000
8000
6000
4000
2000
0
40
60
80
100
120


Age
14000
12000
10000
8000
6000
4000
2000
0
20
30
40
50
60
70



In [14]:
```
df.head()
```

Out[14]:

	purch_amt	gender	card_on_file	age	days_since_last_purch	loyalty
1	65.16	male	yes	23.0	61.0	False
2	40.60	female	no	36.0	49.0	False
3	38.01	male	yes	47.0	57.0	False
6	43.96	male	yes	36.0	64.0	False
9	93.63	female	no	40.0	47.0	True

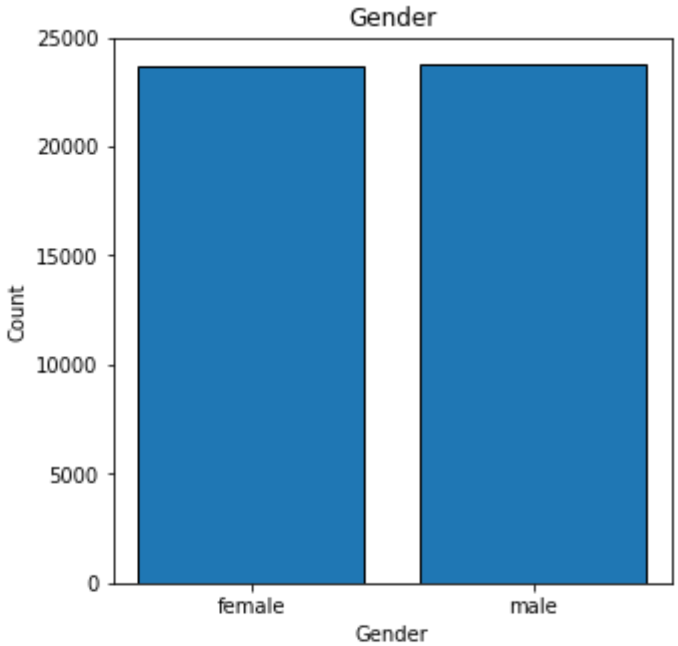
In [15]:
```
# from scipy.stats import boxcox

# df.purch_amt = boxcox(df.purch_amt)[0]
# df.age = boxcox(df.age)[0]
# df.days_since_last_purch = boxcox(df.days_since_last_purch)[0]
```

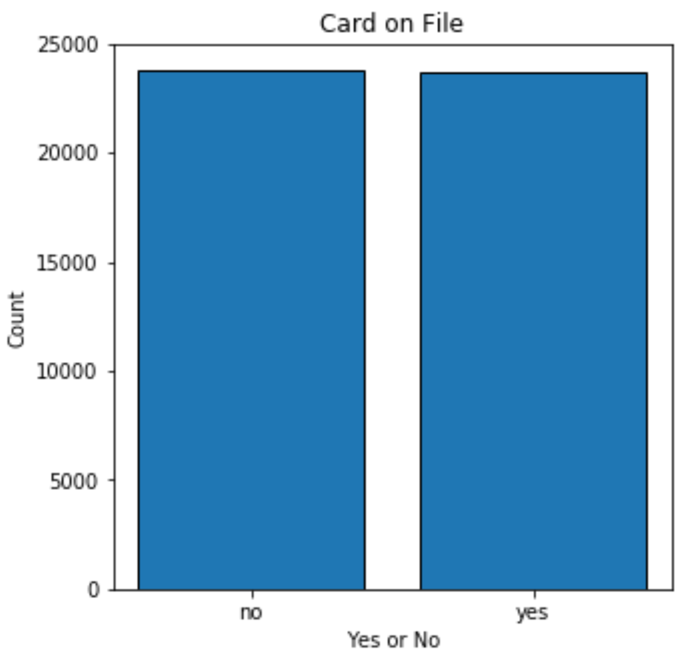
In [16]:
```
df.head()
```

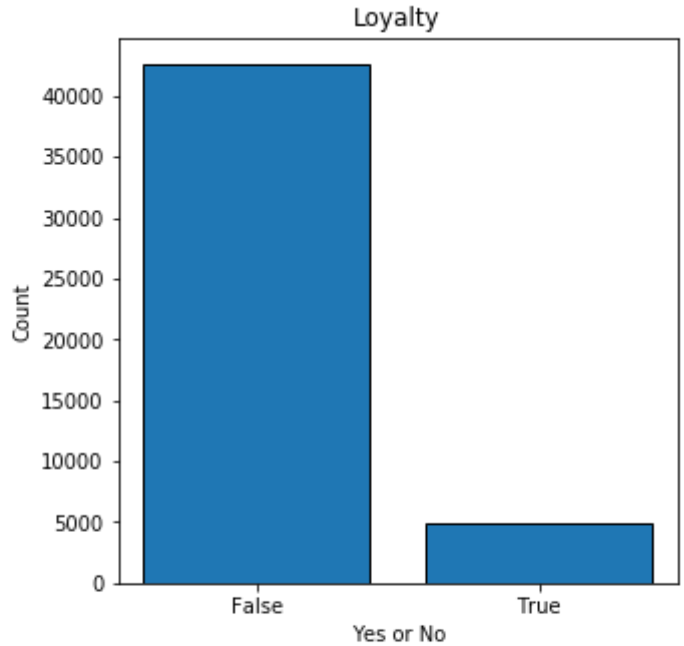
Out[16]:

	purch_amt	gender	card_on_file	age	days_since_last_purch	loyalty
1	65.16	male	yes	23.0	61.0	False
2	40.60	female	no	36.0	49.0	False
3	38.01	male	yes	47.0	57.0	False
6	43.96	male	yes	36.0	64.0	False
9	93.63	female	no	40.0	47.0	True

In [17]:
```
# Visualize categorical variables, class balances

x = ['female', 'male']
y = list(df.groupby(['gender']).count().loyalty)

plt.figure(figsize=(5,5))
plt.bar(x,y, edgecolor = 'k')
plt.title('Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

x = ['no', 'yes']
y = list(df.groupby(['card_on_file']).count().loyalty)

plt.figure(figsize=(5,5))
plt.bar(x,y, edgecolor = 'k')
plt.title('Card on File')
plt.xlabel('Yes or No')
plt.ylabel('Count')
plt.show()

x = ['False', 'True']
y = list(df.groupby(['loyalty']).count().gender)

plt.figure(figsize=(5,5))
plt.bar(x,y, edgecolor = 'k')
plt.title('Loyalty')
plt.xlabel('Yes or No')
plt.ylabel('Count')
plt.show()
```


Gender
25000
20000
15000
10000
5000
0
Count
female
male
Gender


Card on File
25000
20000
15000
10000
5000
0
Count
no
yes
Yes or No



Correct Class Imbalance

In [18]:
```
# Correct Class Imbalance by Under Sampling "No Loyalty"
no_loyalty = df[df['loyalty'] == False]
no_loyalty = no_loyalty.sample(frac=.115)

loyalty = df[df['loyalty'] == True]

undersample = pd.concat([no_loyalty, loyalty])
undersample = undersample.reset_index()
undersample = undersample.drop(['index'], axis=1)
```

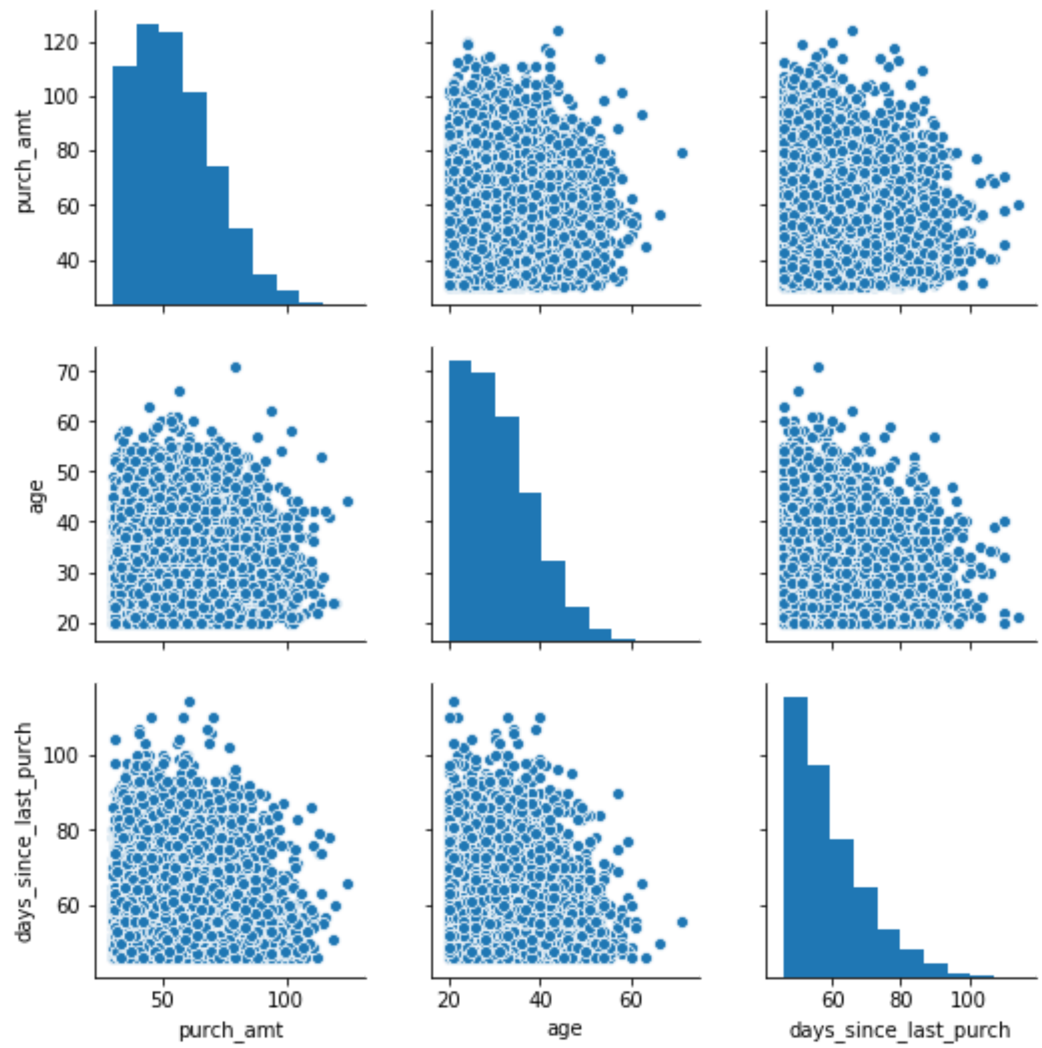
In [19]:
```
undersample.shape
```

Out[19]:
```
(9741, 6)
```

In [20]:
```
# Visualize Class Balabnce of Loyalty
x = ['False', 'True']
y = list(undersample.groupby(['loyalty']).count().gender)

plt.figure(figsize=(6,6))
plt.bar(x,y, edgecolor = 'k')
plt.title('Loyalty')
plt.xlabel('Yes or No')
plt.ylabel('Count')
plt.show()
```

In [21]:
```
# Visualize Correlation and Distributions of undersampled dataset
sns.pairplot(undersample[['purch_amt', 'age', 'days_since_last_purch']])
```

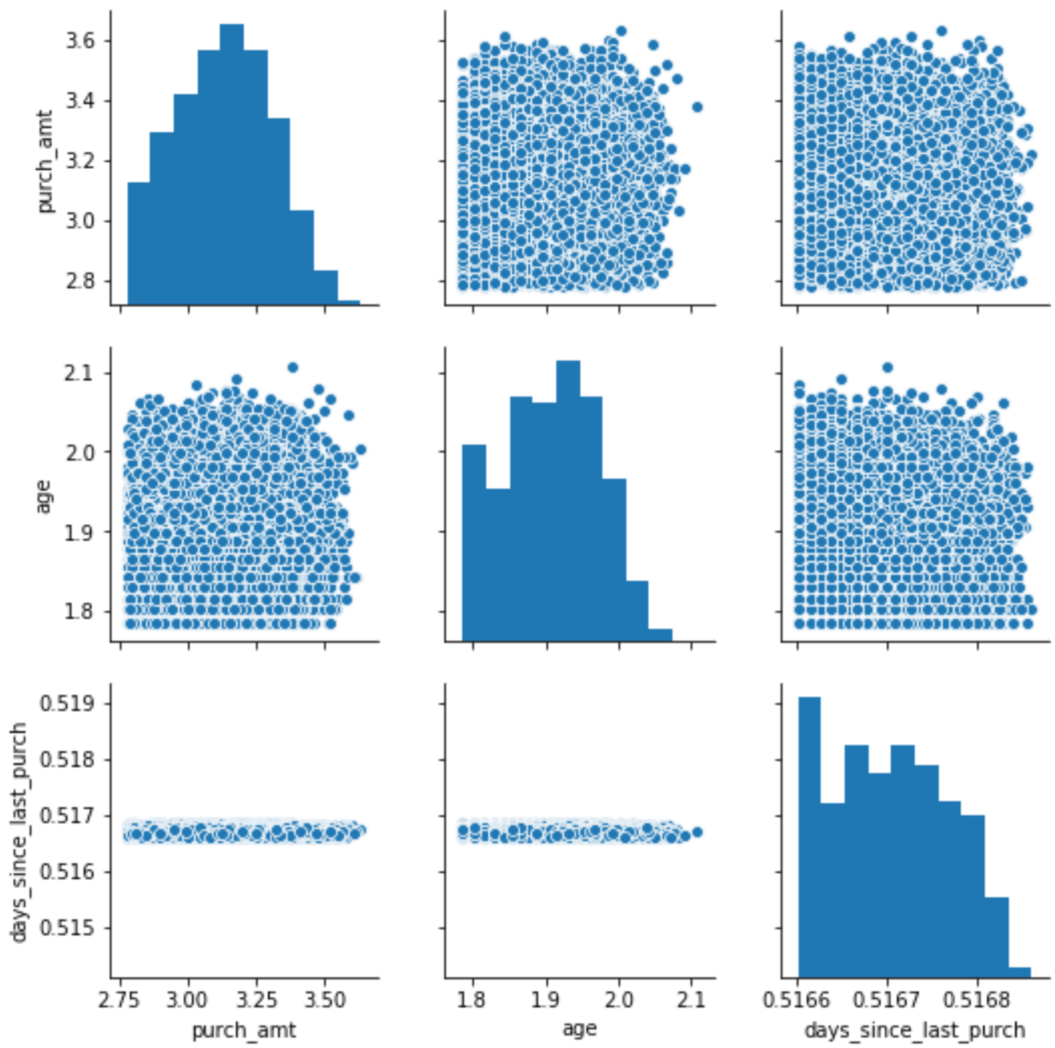
Out[21]: <seaborn.axisgrid.PairGrid at 0x10f935b38>



In [22]:
```
# Correct Right Skew distribution with Box Box Distribution
undersample.purch_amt = boxcox(undersample.purch_amt)[0]
undersample.age = boxcox(undersample.age)[0]
undersample.days_since_last_purch = boxcox(undersample.days_since_last_p
urch)[0]
```

In [23]:
```
# Visualize Distributions after BoxCox Transformation
sns.pairplot(undersample[['purch_amt', 'age', 'days_since_last_purch']])
```

Out[23]: <seaborn.axisgrid.PairGrid at 0x10f911e10>



In [24]:
```
# Turn categorical Variables in to Numeric
undersample = pd.get_dummies(undersample, columns = ['gender', 'card_on_
file'])
undersample['loyalty'] = np.where(undersample['loyalty']==True, 1, 0)
```

In [25]:
```
undersample.head()
```

Out[25]:

	purch_amt	age	days_since_last_purch	loyalty	gender_female	gender_male	card_on_file_
0	3.067936	1.887157	0.516721	0	0	1	
1	3.273919	1.954479	0.516715	0	0	1	
2	2.930705	1.906285	0.516744	0	1	0	
3	3.410857	1.923729	0.516764	0	1	0	
4	3.417293	1.816201	0.516648	0	1	0	

Baseline Models

In [26]:
```
# Split into test and training set
X = undersample.drop(['loyalty'], axis=1)
Y = undersample['loyalty']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=0)

# List of models
models = [RandomForestClassifier(n_estimators=200, max_depth=3, random_s
tate=42), LinearSVC(),
          GaussianNB(), LogisticRegression(random_state=42), GradientBoo
stingClassifier(n_estimators=200, random_state=42)]

model_names = []
accuracies = []

# Iterate through models to compare accuracies
for model in models:
    model_name = model.__class__.__name__
    model_names.append(model_name)
    model.fit(x_train, y_train)
    predictions = model.score(x_train, y_train)
    accuracies.append(predictions)


# Assign accuracies for each model to view
accuracy_df = pd.DataFrame()
accuracy_df['Model'] = model_names
accuracy_df['Accuracy'] = accuracies
```

```
/usr/local/lib/python3.6/site-packages/sklearn/linear_model/logistic.p
y:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
```

In [27]:
```
accuracy_df
```

Out[27]:

	Model	Accuracy
0	RandomForestClassifier	0.724974
1	LinearSVC	0.615375
2	GaussianNB	0.733958
3	LogisticRegression	0.615503
4	GradientBoostingClassifier	0.761037

In [28]:
```
# GradientBoostingClassifier cross validation

gbc = GradientBoostingClassifier(n_estimators=200, random_state=42)
gbc.fit(x_train, y_train)
y_pred = gbc.predict(x_test)
cross_val_score(gbc, X, Y, cv=5)
```

Out[28]:
```
array([0.73217034, 0.74332649, 0.73767967, 0.73767967, 0.71252567])
```

In [29]:
```
# View accuracy scores on classifying each author (precission, recall, f
1-score and support)
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.72      0.73       957
           1       0.74      0.77      0.75       992

   micro avg       0.74      0.74      0.74      1949
   macro avg       0.74      0.74      0.74      1949
weighted avg       0.74      0.74      0.74      1949
```

In [30]:
```
# Pass linear regression model to the RFE constructor

selector = RFE(gbc)
selector = selector.fit(X, Y)

# Sort ranked features
pd.set_option("display.max_rows", 999)
rankings = pd.DataFrame({'Features': X.columns, 'Ranking' : selector.ran
king_})
rankings = rankings.sort_values('Ranking')
rankings = rankings.reset_index()
rankings = rankings.drop(columns=['index'])
rankings = rankings.set_index('Features')
rankings = rankings.T
rankings
```

Out[30]:

Features	purch_amt	age	days_since_last_purch	gender_female	card_on_file_yes	gender_male
Ranking	1	1	1	2	3	4

In [31]:
```
# RandomForestClassifier cross validation

rfc = RandomForestClassifier(n_estimators=300, max_depth=2, random_state
=42)
rfc.fit(x_train, y_train)
y_pred = rfc.predict(x_test)
cross_val_score(rfc, X, Y, cv=5)
```

Out[31]:
```
array([0.72036942, 0.7325462 , 0.71355236, 0.72638604, 0.70328542])
```

In [32]:
```
# View accuracy scores on classifying each author (precission, recall, f
1-score and support)
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.62      0.69       957
           1       0.69      0.83      0.76       992

   micro avg       0.73      0.73      0.73      1949
   macro avg       0.74      0.73      0.72      1949
weighted avg       0.74      0.73      0.72      1949
```

In [33]:
```
# Pass linear regression model to the RFE constructor

selector = RFE(rfc)
selector = selector.fit(X, Y)

# Sort ranked features
pd.set_option("display.max_rows", 999)
rankings = pd.DataFrame({'Features': X.columns, 'Ranking' : selector.ran
king_})
rankings = rankings.sort_values('Ranking')
rankings = rankings.reset_index()
rankings = rankings.drop(columns=['index'])
rankings = rankings.set_index('Features')
rankings = rankings.T
rankings
```

Out[33]:

Features	purch_amt	age	days_since_last_purch	card_on_file_yes	card_on_file_no	gender_femal
Ranking	1	1	1	2	3	4

Feature Engineering

In [34]:
```
# Feature Engineering
undersample['purch_amt'] = undersample['purch_amt'].apply(lambda x: x**2
)
undersample = undersample.drop(['card_on_file_yes', 'gender_male'], axis
=1)
```

Second Round Training on New Features and Predictions

In [35]:
```
# GradientBoostingClassifier cross validation

gbc = GradientBoostingClassifier(n_estimators=200, random_state=42)
gbc.fit(x_train, y_train)
y_pred = gbc.predict(x_test)
cross_val_score(gbc, X, Y, cv=5)
```

Out[35]:
```
array([0.73217034, 0.74332649, 0.73767967, 0.73767967, 0.71252567])
```

In [36]:
```
# View accuracy scores on classifying each author (precission, recall, f
1-score and support)
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.72      0.73       957
           1       0.74      0.77      0.75       992

   micro avg       0.74      0.74      0.74      1949
   macro avg       0.74      0.74      0.74      1949
weighted avg       0.74      0.74      0.74      1949
```

In [37]:
```
# RandomForestClassifier cross validation

rfc = RandomForestClassifier(n_estimators=300, max_depth=2, criterion='e
ntropy', random_state=42)
rfc.fit(x_train, y_train)
y_pred = rfc.predict(x_test)
cross_val_score(rfc, X, Y, cv=5)
```

Out[37]:
```
array([0.71626475, 0.73203285, 0.71201232, 0.72279261, 0.7073922 ])
```

In [38]:
```
# View accuracy scores on classifying each author (precission, recall, f
1-score and support)
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.61      0.69       957
           1       0.69      0.85      0.76       992

   micro avg       0.73      0.73      0.73      1949
   macro avg       0.74      0.73      0.72      1949
weighted avg       0.74      0.73      0.72      1949
```

In [ ]:
```

```