Predicting a Binary Classifier

The purpose of this project is to create a machine learning model to predict whether entries in the dataset belong to a specific class (`'y'`). This dataset was provided by a client as skills assessment, and the objectives and guidelines for the project were outlined by the client in advance. The steps required to complete the project will be:

- Data Cleansing
- Feature Engineering and Selection
- Building two different models on training dataset
- Model tuning
- Generating predictions for test dataset
- Comparing modeling approaches in a writeup
- Preparing all files for submission

Initial Analysis and Data Cleansing

To begin, I will import the `'training'` dataset into a pandas dataframe and perform initial analysis to determine what cleansing steps need to be taken.

In [2]:
```
# Importing necessary data manipulation/visualization libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# Expanding number of rows and columns allowed in view
pd.set_option('max_columns', 180)
pd.set_option('max_rows', 200000)
pd.set_option('max_colwidth', 5000)
```

In [3]:
```
# Reading training and test data into dataframes
train = pd.read_csv('traindata.csv')
test = pd.read_csv('testdata.csv')
```

In [4]:
```
# Exploring shape and head of test data
print(test.shape)
test.head()
```

```
(10000, 100)
```

Out[4]:

	x0	x1	x2	x3	x4	x5	x6	x7
0	0.519093	-4.606038	13.707586	-17.990903	12.873394	14.910935	2.915341	-10.110081
1	-12.357004	13.874141	14.052924	34.129247	34.511107	34.583336	-0.482540	-6.583407
2	1.834922	2.665252	-44.873210	21.941920	10.102981	5.962249	-5.733909	-4.061670
3	20.972483	11.548506	-40.924625	-35.296796	-35.253101	-14.601890	5.045075	10.841771
4	-9.916044	5.509811	31.749288	-0.803916	-4.005098	20.912490	0.419346	-2.949516

In [5]:
```
# Exploring shape and head of train data
print(train.shape)
train.head()
```

```
(40000, 101)
```

Out[5]:

	x0	x1	x2	x3	x4	x5	x6	x7	
0	0.963686	6.627185	-45.224008	9.477531	-3.216532	13.216874	9.754747	5.245851	-1
1	-1.770062	-23.610459	-0.964003	-31.981497	-10.294599	-10.240251	-1.518888	-1.675208	0
2	9.962401	-8.349849	23.248891	-24.196879	8.937480	10.965000	-7.490596	-3.025094	0
3	-5.780709	-25.261584	1.383115	-11.786929	7.993078	-11.245752	-2.607351	-3.513896	-0
4	1.211541	1.119963	7.512938	21.987312	-5.155392	10.339416	3.045180	-0.619230	-0

In [6]:
```
# Analyzing quick statistics on distribution of each feature
train.describe()
```

Out[6]:

	x0	x1	x2	x3	x4	x5	
count	39988.000000	39990.000000	39993.000000	39987.000000	39993.000000	39992.000000	399
mean	2.020255	-3.924559	1.006619	-1.378330	0.070199	-0.715213	
std	9.590599	18.768656	21.062970	29.397779	20.243287	18.268807	
min	-36.842503	-79.156374	-89.728356	-126.652341	-76.412886	-73.743342	-
25%	-4.461433	-16.591552	-13.230956	-21.297149	-13.580632	-13.092873	
50%	2.022412	-4.061703	1.184946	-1.224625	0.091600	-0.657601	
75%	8.389979	8.529110	15.221205	18.530623	13.722427	11.610239	
max	44.478690	77.682652	84.625640	117.004453	85.934044	74.465465	

In [7]:
```
# Looking at types of each feature
train.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 101 columns):
x0     float64
x1     float64
x2     float64
x3     float64
x4     float64
x5     float64
x6     float64
x7     float64
x8     float64
x9     float64
x10    float64
x11    float64
x12    float64
x13    float64
x14    float64
x15    float64
x16    float64
x17    float64
x18    float64
x19    float64
x20    float64
x21    float64
x22    float64
x23    float64
x24    float64
x25    float64
x26    float64
x27    float64
x28    float64
x29    float64
x30    float64
x31    float64
x32    float64
x33    float64
x34    object
x35    object
x36    float64
x37    float64
x38    float64
x39    float64
x40    float64
x41    object
x42    float64
x43    float64
x44    float64
x45    object
x46    float64
x47    float64
x48    float64
x49    float64
x50    float64
x51    float64
x52    float64
x53    float64
```

```
x54    float64
x55    float64
x56    float64
x57    float64
x58    float64
x59    float64
x60    float64
x61    float64
x62    float64
x63    float64
x64    float64
x65    float64
x66    float64
x67    float64
x68    object
x69    float64
x70    float64
x71    float64
x72    float64
x73    float64
x74    float64
x75    float64
x76    float64
x77    float64
x78    float64
x79    float64
x80    float64
x81    float64
x82    float64
x83    float64
x84    float64
x85    float64
x86    float64
x87    float64
x88    float64
x89    float64
x90    float64
x91    float64
x92    float64
x93    object
x94    float64
x95    float64
x96    float64
x97    float64
x98    float64
x99    float64
y      int64
dtypes: float64(94), int64(1), object(6)
memory usage: 30.8+ MB
```

In [8]:
```
# Checking null counts of each feature
nullcounts = train.isnull().sum()
nullcounts.sort_values(ascending=False, inplace=True)
nullcounts
```

Out[8]:
```
x55    16
x13    15
x18    14
x96    14
x3     13
x99    13
x51    13
x62    13
x73    13
x21    12
x85    12
x0     12
x60    12
x56    11
x24    11
x17    11
x33    11
x69    11
x42    11
x12    11
x77    11
x65    10
x7     10
x59    10
x39    10
x35    10
x26    10
x1     10
x66    10
x89    10
x75    10
x28     9
x11     9
x23     9
x46     9
x94     9
x6      9
x87     9
x58     9
x97     9
x31     9
x68     9
x72     9
x74     9
x19     8
x15     8
x78     8
x5      8
x48     8
x82     8
x16     8
x34     8
x63     8
x40     8
x76     8
x57     8
x25     7
```

```
x90      7
x80      7
x4       7
x45      7
x79      7
x10      7
x27      7
x95      7
x52      7
x93      7
x61      7
x86      7
x2       7
x67      7
x9       7
x22      6
x70      6
x36      6
x38      6
x71      6
x8       6
x92      6
x54      6
x14      5
x50      5
x53      5
x88      5
x37      5
x64      5
x20      4
x98      4
x83      4
x49      4
x41      4
x47      3
x29      3
x44      3
x81      3
x84      3
x32      3
x91      2
x30      2
x43      1
y        0
dtype: int64
```

In [9]:

```
# Printing total of all null values
print(nullcounts.sum())
nullpct = nullcounts.sum()/len(train)
print(nullpct)
```

```
806
0.02015
```

Initial Observations

After a quick exploration of the training dataset, a few key observations stand out:

- The data is comprised of 40,000 entries, with 100 features.
- A small portion of the features are string objects, with the rest stored as numeric (float) features
- The features apppear to be of different ranges and sizes, but based on the min and max values and standard deviations there do not appear to be any huge outliers.
- Each feature has some null values, with only 16 in any one column and 806 null values overall. This is a small percentage of the overall dataset (2.01% of total data).
- The `'y'`, or target, column is binary. Each entry is listed as a `0` or `1` in this column, and there are no null values.
- The target column is unbalanced. Approximately 20% of the data is categorized as a `'1'` and 80% as a `'0'`. I will need to account for this imbalance in the machine learning models.

Cleansing Steps Needed

To prepare the data for model training and testing, I need to cleanse the data. This will include:

- Examining and standardizing the text-based features
- Converting text-based features to dummy variables
- Handling null values for each feature
- Standardizing all columns with a min-max scaler for efficient machine learning input

Note: Because these same processes will need to be performed on the testing dataset, the data cleansing will need to be incorporated into a function that can be applied to the test dataset at a later point.

Data Cleansing

Text-Based Features

In [10]:
```
# Identifying text columns
text_vals = train.select_dtypes(include=['object'])
text_vals.head()
```

Out[10]:

	x34	x35	x41	x45	x68	x93
0	chrystler	thur	$-865.28	0.02%	sept.	asia
1	volkswagon	thur	$325.27	-0.01%	July	asia
2	bmw	thurday	$743.91	0.0%	July	asia
3	nissan	thurday	$538.48	0.01%	July	asia
4	volkswagon	wed	$-433.65	0.0%	Jun	asia

In [11]:
```
# Cleaning the values of x41 and x45, storing as float
text_vals['x41'] = text_vals['x41'].str.replace('$','').astype('float')
text_vals['x45'] = text_vals['x45'].str.replace('%','').astype('float')
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy

/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imp
orts until
```

In [12]:
```
# Checking value counts of x34
text_vals['x34'].value_counts(sort=True)
```

Out[12]:
```
volkswagon    12622
Toyota        10968
bmw            7262
Honda          5174
tesla          2247
chrystler      1191
nissan          326
ford            160
mercades         31
chevrolet        11
Name: x34, dtype: int64
```

These values seem to be fairly standardized, and are not significantly skewed. Several contain spelling errors which can be quickly cleaned with a mapping dictionary.

In [13]:
```
# Creating a mapping dictionary for correct labeling
dict = {'volkswagon': 'volkswagen', 'Toyota':'toyota',
       'bmw':'bmw', 'Honda':'honda', 'tesla':'tesla',
        'chrystler':'chrysler', 'nissan':'nissan',
        'ford':'ford','mercades':'mercedes',
       'chevrolet':'chevrolet'}
# Replacing values using dictionary
text_vals['x34'] = text_vals['x34'].replace(dict)
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
```

In [14]:
```
# Checking value counts of x35
text_vals['x35'].value_counts(sort=True)
```

Out[14]:
```
wed          14820
thurday      13324
wednesday     5938
thur          4428
tuesday        884
friday         517
monday          53
fri             26
Name: x35, dtype: int64
```

Again, these are not significantly skewed, but several are misspelled and miscategorized. This can be quickly cleaned.

In [15]:
```
# Creating mapping dictionary
dict = {'wed':'wed', 'thurday':'thu',
       'wednesday':'wed','thur':'thu',
        'tuesday':'tue','friday':'fri',
        'monday':'mon','fri':'fri'}
# Replacing values using dictionary
text_vals['x35'] = text_vals['x35'].replace(dict)
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  import sys
```

In [16]:
```
# Checking value counts of x68
text_vals['x68'].value_counts(sort=True)
```

Out[16]:
```
July       11114
Jun         9317
Aug         8170
May         4744
sept.       3504
Apr         1629
Oct          885
Mar          407
Nov          145
Feb           48
Dev           16
January       12
Name: x68, dtype: int64
```

These can again be cleaned with simple mapping. The one exception is the `"Dev"` value, which I am assuming is a misspelling of December.

In [17]:
```
# Creating mapping dictionary
dict = {'July':'jul','Jun':'jun',
       'Aug':'aug','May':'may',
       'sept.':'sep','Apr':'apr',
       'Oct':'oct','Mar':'mar',
       'Nov':'nov','Feb':'feb',
       'Dev':'dec','January':'jan'}
# Replacing values using dictionary
text_vals['x68'] = text_vals['x68'].replace(dict)
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

In [18]:
```
# Checking value counts of x93
text_vals['x93'].value_counts(sort=True)
```

Out[18]:
```
asia       35384
america     3167
euorpe      1442
Name: x93, dtype: int64
```

This feature seems to be heavily weighted in one category `'asia'`, representing 88% of the total entries. This kind of heavily skewed feature can be problematic for the machine learning model, so this feature should be dropped.

In [19]:
```
# Drop x93 from text vals dataframe
text_vals = text_vals.drop(columns='x93')
```

In [20]:
```
text_vals.head()
```

Out[20]:

	x34	x35	x41	x45	x68
0	chrysler	thu	-865.28	0.02	sep
1	volkswagen	thu	325.27	-0.01	jul
2	bmw	thu	743.91	0.00	jul
3	nissan	thu	538.48	0.01	jul
4	volkswagen	wed	-433.65	0.00	jun

Now that I have cleaned and standardized the text columns, I want to transform them into dummy categories to ensure the machine learning model is operating on stricly numeric data.

In [21]:
```
# Selecting only text columns
textcols = ['x34','x35','x68']
# Get dummy prefix names
prefixes = list(text_vals[textcols].columns)
# Getting dummy features for each text column
text_dummies = pd.get_dummies(text_vals, prefix=prefixes)
```

In [22]:
```
# Dropping dirty text columns from original dataset
dropcols = ['x34','x35','x41','x45','x68','x93']
train = train.drop(columns=dropcols)
```

In [23]:
```
# Reintroducing our cleaned data into the original dataset
train = pd.concat([train,text_dummies], axis=1)
```

In [24]:
```
train.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 124 columns):
x0                  float64
x1                  float64
x2                  float64
x3                  float64
x4                  float64
x5                  float64
x6                  float64
x7                  float64
x8                  float64
x9                  float64
x10                 float64
x11                 float64
x12                 float64
x13                 float64
x14                 float64
x15                 float64
x16                 float64
x17                 float64
x18                 float64
x19                 float64
x20                 float64
x21                 float64
x22                 float64
x23                 float64
x24                 float64
x25                 float64
x26                 float64
x27                 float64
x28                 float64
x29                 float64
x30                 float64
x31                 float64
x32                 float64
x33                 float64
x36                 float64
x37                 float64
x38                 float64
x39                 float64
x40                 float64
x42                 float64
x43                 float64
x44                 float64
x46                 float64
x47                 float64
x48                 float64
x49                 float64
x50                 float64
x51                 float64
x52                 float64
x53                 float64
x54                 float64
x55                 float64
x56                 float64
x57                 float64
```

```
x58              float64
x59              float64
x60              float64
x61              float64
x62              float64
x63              float64
x64              float64
x65              float64
x66              float64
x67              float64
x69              float64
x70              float64
x71              float64
x72              float64
x73              float64
x74              float64
x75              float64
x76              float64
x77              float64
x78              float64
x79              float64
x80              float64
x81              float64
x82              float64
x83              float64
x84              float64
x85              float64
x86              float64
x87              float64
x88              float64
x89              float64
x90              float64
x91              float64
x92              float64
x94              float64
x95              float64
x96              float64
x97              float64
x98              float64
x99              float64
y                int64
x41              float64
x45              float64
x34_bmw          uint8
x34_chevrolet    uint8
x34_chrysler     uint8
x34_ford         uint8
x34_honda        uint8
x34_mercedes     uint8
x34_nissan       uint8
x34_tesla        uint8
x34_toyota       uint8
x34_volkswagen   uint8
x35_fri          uint8
x35_mon          uint8
x35_thu          uint8
x35_tue          uint8
```

```
x35_wed           uint8
x68_apr           uint8
x68_aug           uint8
x68_dec           uint8
x68_feb           uint8
x68_jan           uint8
x68_jul           uint8
x68_jun           uint8
x68_mar           uint8
x68_may           uint8
x68_nov           uint8
x68_oct           uint8
x68_sep           uint8
dtypes: float64(96), int64(1), uint8(27)
memory usage: 30.6 MB
```

We now have a dataframe including only numeric values.

Handling Missing Values

I will now correct any missing values. The amount of missing data is small in proportion to the overall dataset. I could easily replace null values with the mean of each feature. However, since I do not know the nature of each feature and what it represents, this could result in distorted, noisy information that skews the overall results of the model. Instead, I will simply drop any rows that contain null values.

In [25]:
```
# Dropping rows with null values
train = train.dropna().copy()
```

In [26]:
```
nullcounts = train.isnull().sum()
nullcounts
```

Out[26]:

```
x0     0
x1     0
x2     0
x3     0
x4     0
x5     0
x6     0
x7     0
x8     0
x9     0
x10    0
x11    0
x12    0
x13    0
x14    0
x15    0
x16    0
x17    0
x18    0
x19    0
x20    0
x21    0
x22    0
x23    0
x24    0
x25    0
x26    0
x27    0
x28    0
x29    0
x30    0
x31    0
x32    0
x33    0
x36    0
x37    0
x38    0
x39    0
x40    0
x42    0
x43    0
x44    0
x46    0
x47    0
x48    0
x49    0
x50    0
x51    0
x52    0
x53    0
x54    0
x55    0
x56    0
x57    0
x58    0
x59    0
x60    0
```

```
x61              0
x62              0
x63              0
x64              0
x65              0
x66              0
x67              0
x69              0
x70              0
x71              0
x72              0
x73              0
x74              0
x75              0
x76              0
x77              0
x78              0
x79              0
x80              0
x81              0
x82              0
x83              0
x84              0
x85              0
x86              0
x87              0
x88              0
x89              0
x90              0
x91              0
x92              0
x94              0
x95              0
x96              0
x97              0
x98              0
x99              0
y                0
x41              0
x45              0
x34_bmw          0
x34_chevrolet    0
x34_chrysler     0
x34_ford         0
x34_honda        0
x34_mercedes     0
x34_nissan       0
x34_tesla        0
x34_toyota       0
x34_volkswagen   0
x35_fri          0
x35_mon          0
x35_thu          0
x35_tue          0
x35_wed          0
x68_apr          0
x68_aug          0
```

```
x68_dec          0
x68_feb          0
x68_jan          0
x68_jul          0
x68_jun          0
x68_mar          0
x68_may          0
x68_nov          0
x68_oct          0
x68_sep          0
dtype: int64
```

In [27]:
```
print(len(train))
print(len(train[train['y']==1]))
```

```
39228
7977
```

This process dropped only 772 overall values, and still left approximately 20% of the `1` values in the target category. This should be more than sufficient for our modeling purposes.

Scaling Features

Machine learning models, especially the classification models I will use on this data, are much more effective on standardized data. To transform this dataset into standardized data, I will use the "minmax" scaler from the scikit-learn library.

In [28]:
```
# Importing sci-kit minmax scaler
from sklearn import preprocessing
```

In [29]:
```
# Saving column names for future dataframe
colnames = train.columns
# Creating minmax scaler instance
mm_scaler = preprocessing.MinMaxScaler()
# Transforming data into scaled array
df_mm = mm_scaler.fit_transform(train)
# Creating new dataframe with scaled data
train_clean = pd.DataFrame(df_mm, columns=colnames)
```

In [30]:
```
train_clean.describe()
```

Out[30]:

	x0	x1	x2	x3	x4	x5	
count	39228.000000	39228.000000	39228.000000	39228.000000	39228.000000	39228.000000	392:
mean	0.477927	0.479577	0.520466	0.514035	0.471174	0.492666	
std	0.117946	0.119690	0.120856	0.120769	0.124693	0.123240	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.398198	0.398858	0.438630	0.432242	0.387015	0.409138	
50%	0.478068	0.478641	0.521456	0.514728	0.471413	0.493099	
75%	0.556340	0.558898	0.601996	0.595674	0.555341	0.575851	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

Now all values have been scaled between the ranges of 0 and 1.

Feature Selection

Now I will narrow down the list of features to help improve the efficiency of the machine learning models. This will involve:

- Checking for correlation with the target column.
- Eliminating features that seem to be unrelated to the target.
- Checking for colinearity to make sure no information is leaking (surving as a proxy for the target).
- Creating a final dataframe for model testing.

In [31]:
```
corr_values = abs(train_clean.corr()['y'])
```

In [32]:
```
corr_values.sort_values(ascending=False, inplace=True)
corr_values
```

Out[32]:
```
y         1.000000
x75       0.204860
x37       0.198846
x97       0.187550
x58       0.184349
x41       0.176463
x70       0.106727
x1        0.104049
x99       0.099482
x22       0.098001
x33       0.096947
x66       0.096597
x79       0.096069
x69       0.095656
x3        0.093992
x21       0.092737
x63       0.092689
x40       0.092125
x78       0.091484
x96       0.091409
x50       0.090401
x83       0.090184
x45       0.089987
x73       0.089966
x2        0.089283
x56       0.088974
x72       0.088764
x5        0.087622
x85       0.086100
x51       0.086080
x10       0.082422
x20       0.080256
x35_thu   0.073902
x0        0.067035
x35_wed   0.058920
x44       0.054281
x35_tue   0.049927
x68_oct   0.026025
x68_apr   0.025676
x68_feb   0.022781
x68_nov   0.022572
x68_jul   0.021731
x35_mon   0.018683
x48       0.013672
x53       0.013388
x68_mar   0.013169
x38       0.012941
x29       0.011586
x68_aug   0.011443
x68_may   0.010876
x9        0.009982
x8        0.009883
x74       0.009397
x68_jun   0.009169
x68_dec   0.008615
x42       0.008502
x68_sep   0.008000
```

```
x88             0.007870
x12             0.007306
x92             0.006818
x98             0.006504
x17             0.006355
x46             0.006313
x6              0.006223
x14             0.006138
x7              0.006066
x82             0.005897
x35_fri         0.005893
x30             0.005721
x16             0.005714
x34_chrysler    0.005634
x13             0.005620
x23             0.005538
x18             0.005336
x65             0.005181
x95             0.005096
x52             0.004992
x34_mercedes    0.004813
x34_chevrolet   0.004679
x54             0.004563
x67             0.004432
x62             0.004286
x80             0.004072
x36             0.004032
x31             0.004019
x25             0.003959
x43             0.003343
x60             0.003246
x94             0.002805
x34_toyota      0.002633
x86             0.002397
x57             0.002364
x34_honda       0.002300
x39             0.002262
x89             0.002231
x68_jan         0.002027
x64             0.002010
x47             0.001993
x76             0.001987
x34_volkswagen  0.001826
x27             0.001739
x34_tesla       0.001663
x19             0.001620
x49             0.001581
x55             0.001411
x81             0.001331
x91             0.001320
x90             0.001218
x4              0.001006
x34_ford        0.000929
x24             0.000896
x32             0.000895
x71             0.000841
x15             0.000820
```

```
x59              0.000712
x34_bmw          0.000710
x11              0.000648
x87              0.000585
x61              0.000513
x34_nissan       0.000336
x26              0.000223
x28              0.000148
x84              0.000028
x77              0.000020
Name: y, dtype: float64
```

It appears as though the values are not stronly correlated with the target variable. (Note: this correlation is being determined with a binary classifier, which is not ideal. Pearson correlation should be used mostly on continuous variables. However, the relative correlation between each feature is what we are interested in, so these results can still be useful for feature selection).

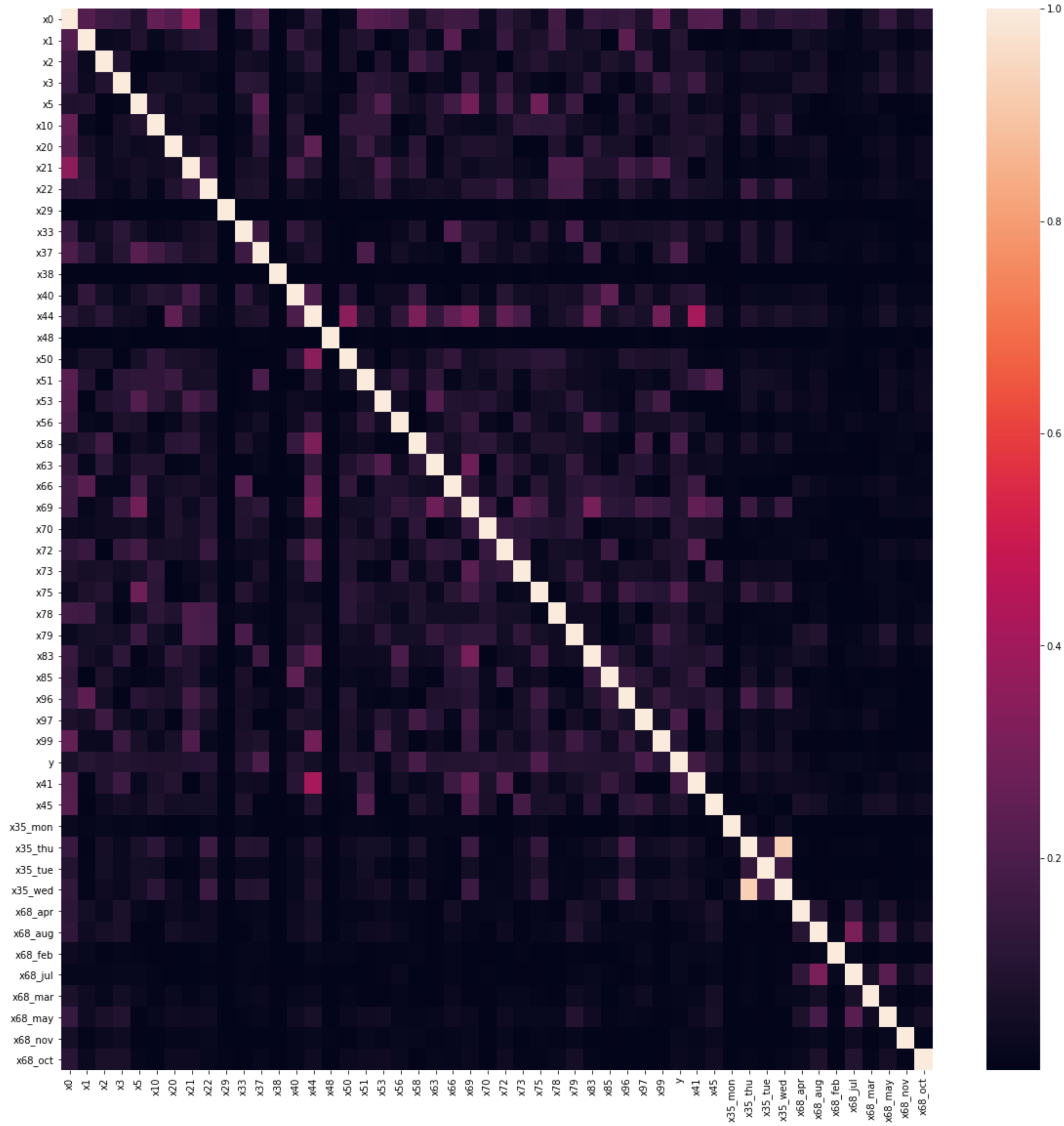
In order to make the model more efficient, I will eleminate any values whose [absolute] correlation is lower than .01

In [33]:
```
# Create list of column names with correlations below .01
low_corr_names = list(corr_values[corr_values <= .01].index)
```

In [34]:
```
# Dropping list of low correlation features from dataset
train_clean = train_clean.drop(columns=low_corr_names)
```

Now I will plot a correlation heatmap to check for colinearity among any of the remaining features

In [35]:
```
# Creating correlation matrix
train_corr = abs(train_clean.corr())
# Mappting correlation matrix
f, ax = plt.subplots(figsize=(20, 20))
ax = sns.heatmap(train_corr)
```

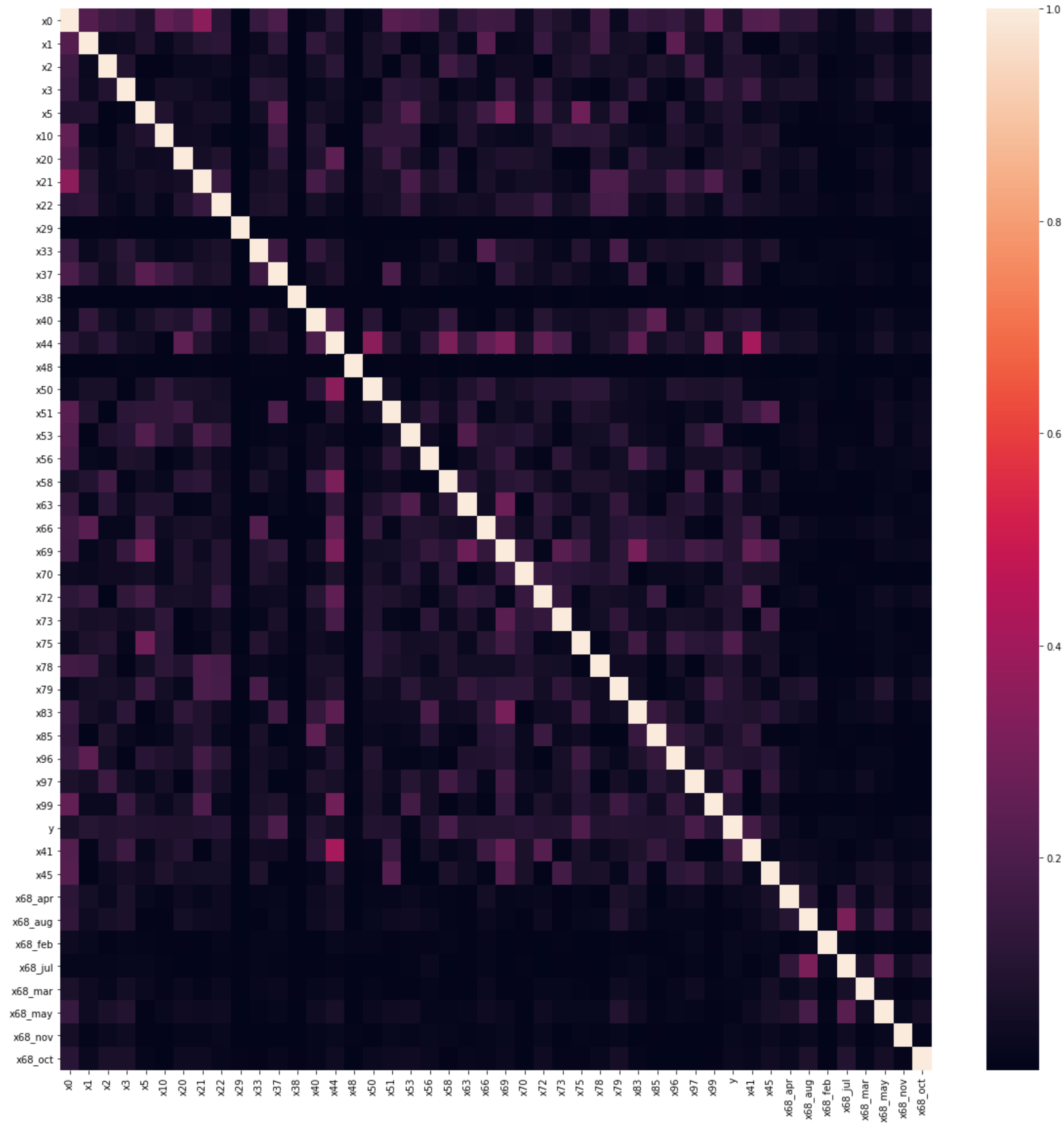


It seems as though the only highly correlated features are `'x35_wed'` and `'x35_thu'`. These are the dummy variables we created above, so they are not simply representing the same data. However, from the value counts we calculated earlier, the wednesday and thursday categories combined account for over 96% of the data. If these two variables are indeed collinear, then they are representing a larger category that is highly skewed in the wed/thu bucket.

Because of this, we should eliminate the `'x35'` feature entirely so it does not throw off the results of the model.

In [36]:
```
# Creating list of x35 categories to drop
dropcols = ['x35_mon','x35_tue','x35_wed','x35_thu']
train_clean = train_clean.drop(columns=dropcols)
```

In [37]:
```
# Recreating correlation matrix
train_corr = abs(train_clean.corr())
# Mappting correlation matrix
f, ax = plt.subplots(figsize=(20, 20))
ax = sns.heatmap(train_corr)
```



It now appears that collinearity has been eliminated and we have a reduced set of features.

Recursive Feature Elimination

To help improve the model even further, I will use the scikit learn library's recursive feature elimination function to help narrow down the number of significant features for training. Since we are attempting to fit a binary classification model, I will implement this feature elimination using logistic regression with cross-fold validation.

In [379]:
```
# Importing feature elimination
from sklearn.feature_selection import RFECV
from sklearn.linear_model import LogisticRegression
```

In [380]:
```
# Creating feature and target sets
all_X = train_clean.drop(columns='y')
all_Y = train_clean['y']
```

In [383]:
```
# Implementing a logistic regression model
lr = LogisticRegression(max_iter=1000)
# Creating a selector for feature elimination
selector = RFECV(lr, cv=10)
# Fitting the selector and model to trianing data
selector.fit(all_X, all_Y)
```

Out[383]:
```
RFECV(cv=10,
      estimator=LogisticRegression(C=1.0, class_weight=None, dual=Fals
e,
                                   fit_intercept=True, intercept_scalin
g=1,
                                   l1_ratio=None, max_iter=1000,
                                   multi_class='auto', n_jobs=None,
                                   penalty='l2', random_state=None,
                                   solver='lbfgs', tol=0.0001, verbose=
0,
                                   warm_start=False),
      min_features_to_select=1, n_jobs=None, scoring=None, step=1, verb
ose=0)
```

In [384]:
```
# Checking for optimized columns
Optimized_columns=all_X.columns[selector.support_]
```

In [387]:
```
print(Optimized_columns)
print(len(Optimized_columns))
```

```
Index(['x0', 'x1', 'x2', 'x3', 'x5', 'x10', 'x20', 'x21', 'x22', 'x29',
'x33',
       'x37', 'x38', 'x40', 'x44', 'x48', 'x50', 'x51', 'x53', 'x56',
'x58',
       'x63', 'x66', 'x69', 'x70', 'x72', 'x73', 'x75', 'x78', 'x79',
'x83',
       'x85', 'x96', 'x97', 'x99', 'x41', 'x45', 'x68_aug', 'x68_feb',
       'x68_mar', 'x68_nov', 'x68_oct'],
      dtype='object')
42
```

In [391]:
```
# finding columns not selected
not_selected=[]
total_cols=list(all_X.columns)
for col in total_cols:
    if col in Optimized_columns:
        pass
    else:
        not_selected.append(col)
```

In [392]:
```
not_selected
```

Out[392]:
```
['x68_apr', 'x68_jul', 'x68_may']
```

After running this optimization, it seems as though the only features which were eliminated were the april, may and june features. We can eliminate these features from the overall dataset and continue with our machine learning process.

In [393]:
```
train_clean = train_clean.drop(columns=not_selected)
```

Model Creation

Error Metric

The error metric I will use for testing the models is provided in the assessment description. Models will be evaluated based on AUC score, so I will use this metric to evaluate the success of each model.

In [404]:
```
# Importing auc metric
from sklearn.metrics import roc_auc_score

# Import Kfold
from sklearn.model_selection import KFold
```

Logistic Regression

To begin the Machine Learning process, I will implement a basic logistic regression model, using the features that remain in the `'train_clean'` dataset. Logistic regression is a great place to start for a binary classifier, and will give us a baseline against which to test any other models for improvement.

First, I will define a function for the logistic regresssion model with K-fold cross validation.

In [418]:
```
# Def a function for training/testing
def train_and_test(df, k=0):
    # Splitting dataframe from target column
    features = df.columns.drop('y')
    lr=LogisticRegression(max_iter=1000)

    # Building K-folds
    kf = KFold(n_splits=k, shuffle=True)
    auc_values = []
    for train_index, test_index, in kf.split(df):
        # Creating train/test set for fold
        train = df.iloc[train_index]
        test = df.iloc[test_index]
        # Fitting and predicting
        lr.fit(train[features], train['y'])
        predictions = lr.predict(test[features])
        # Calculate AUC
        auc = roc_auc_score(test['y'], predictions)
        auc_values.append(auc)
    # Averaging auc values
    avg_auc = np.mean(auc_values)
    var_auc = np.var(auc_values)
    print(avg_auc, var_auc)
    return avg_auc, var_auc
```

Next, I will train the model with a variety of K-fold values to see which provides the best accuracy.

In [415]:
```
# Creating a list of k-fold values
splits = [2, 4, 6, 8, 10]

# Creating a dictionary of values for auc and var
auc_dict = {}

# Training a model with training data
for i in splits:
    k_auc, k_var = train_and_test(train, i)
    auc_dict[i] = k_auc, k_var
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.7864077223358682 1.445338211400935e-07
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.7863791958247519 2.2179188732573277e-05
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.784796173973918 5.206692782180382e-05
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
0.7863007769084782 0.00011849641462344976
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.7853201730657504 0.00011457379001719168
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [417]:
```
auc_dict
```

Out[417]:
```
{2: (0.7864077223358682, 1.445338211400935e-07),
 4: (0.7863791958247519, 2.2179188732573277e-05),
 6: (0.784796173973918, 5.206692782180382e-05),
 8: (0.7863007769084782, 0.00011849641462344976),
 10: (0.7853201730657504, 0.00011457379001719168)}
```

It appears as though the first iteration, with only 2 folds for cross validation, provided the highest AUC score and lowest variance in AUC scores.

Balanced Class Adjustment

To establish a true baseline, however, I need to run this test again using a 'balanced' class model. Since the 1 y-values are disproportionate to the 0 values, this will help balance out the classes and hopefully improve the accuracy of our model.

In [419]:

```
# Redefining train/test function for balanced class
def train_and_test(df, k=0):
    # Splitting dataframe from target column
    features = df.columns.drop('y')
    # Including balanced class
    lr=LogisticRegression(max_iter=1000, class_weight='balanced')

    # Building K-folds
    kf = KFold(n_splits=k, shuffle=True)
    auc_values = []
    for train_index, test_index, in kf.split(df):
        # Creating train/test set for fold
        train = df.iloc[train_index]
        test = df.iloc[test_index]
        # Fitting and predicting
        lr.fit(train[features], train['y'])
        predictions = lr.predict(test[features])
        # Calculate AUC
        auc = roc_auc_score(test['y'], predictions)
        auc_values.append(auc)
    # Averaging auc values
    avg_auc = np.mean(auc_values)
    var_auc = np.var(auc_values)
    print(avg_auc, var_auc)
    return avg_auc, var_auc
```

In [420]:
```
# Re-running model with balanced classes
# Creating a list of k-fold values
splits = [2, 4, 6, 8, 10]

# Creating a dictionary of values for auc and var
auc_dict = {}

# Training a model with training data
for i in splits:
    k_auc, k_var = train_and_test(train, i)
    auc_dict[i] = k_auc, k_var
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.829410606406946 2.219268039043902e-11
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.8284743020457778 1.9726083728891866e-05
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.8295695189795288 1.3648832633338154e-05
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.8293396643716002 3.137504571509923e-05
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.8288388692373765 5.0362273750487695e-05
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/sklearn/line
ar_model/_logistic.py:939: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [421]:
```python
auc_dict
```

Out[421]:
```
{2: (0.829410606406946, 2.219268039043902e-11),
 4: (0.8284743020457778, 1.9726083728891866e-05),
 6: (0.8295695189795288, 1.3648832633338154e-05),
 8: (0.8293396643716002, 3.137504571509923e-05),
 10: (0.8288388692373765, 5.0362273750487695e-05)}
```

It now appears that our best model used 6 K-folds, returning the highest AUC score and still maintaining a low variance. This will be our threshold for improving model accuracy.

Next I will try a K Nearest Neighbors model.

K Nearest Neighbors & Hyperparameter Tuning

K Nearest Neighbors is an effective tool for binary classification models. Here I will implement a basic model and include some hyperparameter tuning through a grid search. This will create several models and evaluate the best hyperparameters to use.

In [423]:
```python
# Importing K Nearest Neighbors model
from sklearn.neighbors import KNeighborsClassifier
# Importing Grid Search
from sklearn.model_selection import GridSearchCV
```

In [424]:
```python
# Creating features and target dataframes
all_X = train_clean.drop(columns='y')
all_Y = train_clean['y']
```

In [427]:
```
# Dictionary of hyperparameters to test
hyperparameters = {
    "n_neighbors": range(1,20,2),
    "weights": ["distance", "uniform"],
    "algorithm": ['brute'],
    "p": [1,2]
}
# model selection
knn = KNeighborsClassifier()
# grid search cv (param=dictionary we created, cv=folds)
grid = GridSearchCV(knn,param_grid=hyperparameters,cv=10)

grid.fit(all_X, all_Y)

# returning best hyperparameters and score
best_params = grid.best_params_
best_score = grid.best_score_

# training using the best model
best_knn = grid.best_estimator_
```

In [429]:
```
best_params
```

Out[429]:
```
{'algorithm': 'brute', 'n_neighbors': 5, 'p': 2, 'weights': 'distance'}
```

In [430]:
```
# Redefining train/test function for optimized knn model
def train_and_test_knn(df, k=0):
    # Splitting dataframe from target column
    features = df.columns.drop('y')
    # Including balanced class
    knn = KNeighborsClassifier(n_neighbors=5, weights='distance', algori
thm=brute, p=2)

    # Building K-folds
    kf = KFold(n_splits=k, shuffle=True)
    auc_values = []
    for train_index, test_index, in kf.split(df):
        # Creating train/test set for fold
        train = df.iloc[train_index]
        test = df.iloc[test_index]
        # Fitting and predicting
        knn.fit(train[features], train['y'])
        predictions = knn.predict(test[features])
        # Calculate AUC
        auc = roc_auc_score(test['y'], predictions)
        auc_values.append(auc)
    # Averaging auc values
    avg_auc = np.mean(auc_values)
    var_auc = np.var(auc_values)
    print(avg_auc, var_auc)
    return avg_auc, var_auc
```

In [431]:
```
# Running model for optimized knn
train_and_test(train_clean, k=10)
```

```
0.814493114647189 5.168933046039276e-05
```

Out[431]:
```
(0.814493114647189, 5.168933046039276e-05)
```

KNN Results

After optimizing a K Nearest Neighbor model, the AOC score returned as .8145. This is close to the score of our best Logistic Regression model, but not quite as good.

Random Forest Classifier

The final basic model I will choose is a Random Forest Classifier. This algorithm is well suited to binary classification, and can eliminate the bias of a single decision tree.

I will use the same Grid Search optimization method to try and find the best hyperparameters for a Random Forest model.

In [61]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [437]:

```
# Dictionary of hyperparameters to test
hyperparameters = {
    "n_estimators": range(100,500,100),
    "max_depth": [10],
    'class_weight':['balanced'],
    'n_jobs':[-1],
    'verbose':[1]
    }
# model selection
rf = RandomForestClassifier()
# grid search cv (param=dictionary we created, cv=folds)
grid = GridSearchCV(rf,param_grid=hyperparameters,cv=10)

grid.fit(all_X, all_Y)

# returning best hyperparameters and score
best_params = grid.best_params_
best_score = grid.best_score_

# training using the best model
best_rf = grid.best_estimator_
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    5.6s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    9.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    6.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    7.0s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    6.4s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    6.8s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.0s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    7.1s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.6s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    7.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
```

```
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.4s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    7.0s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    6.4s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    6.4s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.6s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   13.1s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.2s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   12.7s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.1s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   12.6s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.7s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.3s
```

```
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   12.9s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent 
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   16.0s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   16.7s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent 
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.0s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.8s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   13.2s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent 
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.2s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   12.7s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent 
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   16.7s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   17.2s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent 
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.4s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.7s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   14.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent 
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
```

```
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    4.4s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.7s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   15.2s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   20.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   15.1s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   23.7s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   15.8s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   24.2s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   15.6s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   29.8s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   17.9s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   26.4s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
```

```
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.8s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   22.8s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.6s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   26.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    4.9s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   16.6s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   24.6s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.5s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.7s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   22.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.0s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.6s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   21.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   15.4s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   30.7s finished
```

```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.3s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   29.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.9s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   28.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.0s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   28.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.1s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   29.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.3s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.6s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   30.1s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
```

```
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.3s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.9s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   29.2s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.8s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   28.9s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.0s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   14.2s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   29.7s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.4s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   15.8s
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed:   31.9s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 400 out of 400 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   15.6s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   23.4s finished
```

In [438]:

```
best_params
```

Out[438]:

```
{'class_weight': 'balanced',
 'max_depth': 10,
 'n_estimators': 300,
 'n_jobs': -1,
 'verbose': 1}
```

In [441]:
```
# Redefining train/test function for optimized random forest model
def train_and_test_rf(df, k=0):
    # Splitting dataframe from target column
    features = df.columns.drop('y')
    # Including balanced class
    rf = RandomForestClassifier(class_weight='balanced', max_depth=10, n
_estimators=300, n_jobs=-1, verbose=1)

    # Building K-folds
    kf = KFold(n_splits=k, shuffle=True)
    auc_values = []
    for train_index, test_index, in kf.split(df):
        # Creating train/test set for fold
        train = df.iloc[train_index]
        test = df.iloc[test_index]
        # Fitting and predicting
        rf.fit(train[features], train['y'])
        predictions = rf.predict(test[features])
        # Calculate AUC
        auc = roc_auc_score(test['y'], predictions)
        auc_values.append(auc)
    # Averaging auc values
    avg_auc = np.mean(auc_values)
    var_auc = np.var(auc_values)
    print(avg_auc, var_auc)
    return avg_auc, var_auc
```

In [442]:
```
# Running model for optimized rf
train_and_test_rf(train_clean, k=10)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   11.2s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   17.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   11.3s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   17.6s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   11.2s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   17.6s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   11.2s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   17.4s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   11.3s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   17.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   11.6s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   17.8s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
```

```
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   13.2s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   20.8s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   17.5s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   27.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.9s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   19.9s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.7s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.0s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   19.2s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s

0.8653249605810925 0.00010615893586465878

[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
```

Out[442]:
```
(0.8653249605810925, 0.00010615893586465878)
```

Random Forest Results

This algorithm produced an AOC score of 0.865, significantly higher than previous models. The variance has increased, but not enough to rule out this model as the most effective so far.

Neural Network Application

Last, I will run two Neural Network models to see if the accuracy improves.

In [67]:
```
# Importing Neural Network Classifier
from sklearn.neural_network import MLPClassifier
```

In [453]:
```
# Redefining neural neetwork train and test function
def train_and_test_nn(df, k=0, n=0):
    # Splitting dataframe from target column
    features = df.columns.drop('y')
    # Including balanced class
    mlp = MLPClassifier(hidden_layer_sizes=(n,), max_iter=500)

    # Building K-folds
    kf = KFold(n_splits=k, shuffle=True)
    auc_values = []
    for train_index, test_index, in kf.split(df):
        # Creating train/test set for fold
        train = df.iloc[train_index]
        test = df.iloc[test_index]
        # Fitting and predicting
        mlp.fit(train[features], train['y'])
        predictions = mlp.predict(test[features])
        # Calculate AUC
        auc = roc_auc_score(test['y'], predictions)
        auc_values.append(auc)
    # Averaging auc values
    avg_auc = np.mean(auc_values)
    var_auc = np.var(auc_values)
    print(avg_auc, var_auc)
    return avg_auc, var_auc
```

In [454]:
```
# Running model for different neuron levels in one hidden layer
neurons = [8, 16, 32]
accuracies = {}
for i in neurons:
    aoc_score, aoc_var = train_and_test_nn(train_clean, k=4, n=i)
    accuracies[i] = aoc_score, aoc_var
```

```
0.8333999923376425 0.0006858904796699593
0.8710044161766354 0.0003664845593469178
0.908420581924953 0.0005757786280987429
```

This increased the accuracty level significantly, but also increased the variance of the accuracy levels in the K-fold testing. This could mean the model is over-fitting, but the variance level is still very small compared to the overall accuracy level.

Finally, I will run a Neural Network model with two hidden layers, with 16 nodes each.

In [455]:
```
# Redefining neural network for 2 hidden layers
def train_and_test_nn2(df, k=0, n=0):
    # Splitting dataframe from target column
    features = df.columns.drop('y')
    # Including balanced class
    mlp = MLPClassifier(hidden_layer_sizes=(n,n), max_iter=500)

    # Building K-folds
    kf = KFold(n_splits=k, shuffle=True)
    auc_values = []
    for train_index, test_index, in kf.split(df):
        # Creating train/test set for fold
        train = df.iloc[train_index]
        test = df.iloc[test_index]
        # Fitting and predicting
        mlp.fit(train[features], train['y'])
        predictions = mlp.predict(test[features])
        # Calculate AUC
        auc = roc_auc_score(test['y'], predictions)
        auc_values.append(auc)
    # Averaging auc values
    avg_auc = np.mean(auc_values)
    var_auc = np.var(auc_values)
    print(avg_auc, var_auc)
    return avg_auc, var_auc
```

In [456]:
```
# Running model for different neuron levels in one hidden layer
neurons = [16]
accuracies = {}
for i in neurons:
    aoc_score, aoc_var = train_and_test_nn2(train_clean, k=4, n=i)
    accuracies[i] = aoc_score, aoc_var
```

```
0.90822683719581 1.79273849200885e-05
```

In [457]:
```
# Running model for different neuron levels in two hidden layers
neurons = [32]
accuracies = {}
for i in neurons:
    aoc_score, aoc_var = train_and_test_nn2(train_clean, k=4, n=i)
    accuracies[i] = aoc_score, aoc_var
```

```
0.9302986826327426 1.603054636250812e-05
```

This model looks to be the best overall, with the highest AOC and one of the lowest variances of the models we have generated.

Generating Predictions

Based on this testing, it seems the two best fitting models are:

- Neural Network Classifier (AOC: 0.9303) **Hyper-parameters: 2 hidden layers, 32 nodes each** K-fold validation: 4 folds
- Random Forest Classifier (AOC: 0.8653) **Hyper-parameters: class_weight: 'balanced', max_depth: 10, n_estimators: 300, n_jobs: -1** K-fold validation: 10 folds

Now I will use these models to generate predictions for the test data and save those predictions in the appropriate format as per the assessment instructions.

Test Data Cleaning

First, I need to clean the test data with the same process used above.

In [48]:
```
# Reimporting test data to clean dataframe
test = pd.read_csv('testdata.csv')
```

In [49]:
```
# Identifying text columns
text_vals = test.select_dtypes(include=['object'])
text_vals.head()
# Cleaning the values of x41 and x45, storing as float
text_vals['x41'] = text_vals['x41'].str.replace('$','').astype('float')
text_vals['x45'] = text_vals['x45'].str.replace('%','').astype('float')
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  """
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
```

In [50]:
```
# Creating a mapping dictionary for correct labeling
dict = {'volkswagon': 'volkswagen', 'Toyota':'toyota',
        'bmw':'bmw', 'Honda':'honda', 'tesla':'tesla',
         'chrystler':'chrysler', 'nissan':'nissan',
         'ford':'ford','mercades':'mercedes',
        'chevrolet':'chevrolet'}
# Replacing values using dictionary
text_vals['x34'] = text_vals['x34'].replace(dict)

# Creating mapping dictionary
dict2 = {'July':'jul','Jun':'jun',
        'Aug':'aug','May':'may',
        'sept.':'sep','Apr':'apr',
        'Oct':'oct','Mar':'mar',
        'Nov':'nov','Feb':'feb',
        'Dev':'dec','January':'jan'}
# Replacing values using dictionary
text_vals['x68'] = text_vals['x68'].replace(dict2)
```

```
/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy

/Users/eddiekirkland/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
```

In [51]:
```
# Drop x93 from text vals dataframe
text_vals = text_vals.drop(columns='x93')
```

In [52]:
```
text_vals.head()
```

Out[52]:

	x34	x35	x41	x45	x68
0	bmw	thurday	107.93	0.00	jun
1	tesla	thurday	-600.43	0.02	may
2	honda	thurday	103.08	-0.00	jun
3	volkswagen	thurday	1518.78	-0.01	sep
4	volkswagen	thurday	-2324.39	-0.00	jun

In [53]:
```
# Selecting only text columns
textcols = ['x34','x35','x68']
# Get dummy prefix names
prefixes = list(text_vals[textcols].columns)
# Getting dummy features for each text column
text_dummies = pd.get_dummies(text_vals, prefix=prefixes)
```

In [54]:
```
# Dropping dirty text columns from original dataset
dropcols = ['x34','x35','x41','x45','x68','x93']
test = test.drop(columns=dropcols)
```

In [55]:
```
# Reintroducing our cleaned data into the original dataset
test = pd.concat([test,text_dummies], axis=1)
```

In [56]:
```
# Filling null values with mean for feature
test.fillna(value=0, inplace=True)
```

In [57]:
```
test.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 126 columns):
x0                  float64
x1                  float64
x2                  float64
x3                  float64
x4                  float64
x5                  float64
x6                  float64
x7                  float64
x8                  float64
x9                  float64
x10                 float64
x11                 float64
x12                 float64
x13                 float64
x14                 float64
x15                 float64
x16                 float64
x17                 float64
x18                 float64
x19                 float64
x20                 float64
x21                 float64
x22                 float64
x23                 float64
x24                 float64
x25                 float64
x26                 float64
x27                 float64
x28                 float64
x29                 float64
x30                 float64
x31                 float64
x32                 float64
x33                 float64
x36                 float64
x37                 float64
x38                 float64
x39                 float64
x40                 float64
x42                 float64
x43                 float64
x44                 float64
x46                 float64
x47                 float64
x48                 float64
x49                 float64
x50                 float64
x51                 float64
x52                 float64
x53                 float64
x54                 float64
x55                 float64
x56                 float64
x57                 float64
```

```
x58             float64
x59             float64
x60             float64
x61             float64
x62             float64
x63             float64
x64             float64
x65             float64
x66             float64
x67             float64
x69             float64
x70             float64
x71             float64
x72             float64
x73             float64
x74             float64
x75             float64
x76             float64
x77             float64
x78             float64
x79             float64
x80             float64
x81             float64
x82             float64
x83             float64
x84             float64
x85             float64
x86             float64
x87             float64
x88             float64
x89             float64
x90             float64
x91             float64
x92             float64
x94             float64
x95             float64
x96             float64
x97             float64
x98             float64
x99             float64
x41             float64
x45             float64
x34_bmw         uint8
x34_chevrolet   uint8
x34_chrysler    uint8
x34_ford        uint8
x34_honda       uint8
x34_mercedes    uint8
x34_nissan      uint8
x34_tesla       uint8
x34_toyota      uint8
x34_volkswagen  uint8
x35_fri         uint8
x35_friday      uint8
x35_monday      uint8
x35_thur        uint8
x35_thurday     uint8
```

```
x35_tuesday       uint8
x35_wed           uint8
x35_wednesday     uint8
x68_apr           uint8
x68_aug           uint8
x68_dec           uint8
x68_feb           uint8
x68_jan           uint8
x68_jul           uint8
x68_jun           uint8
x68_mar           uint8
x68_may           uint8
x68_nov           uint8
x68_oct           uint8
x68_sep           uint8
dtypes: float64(96), uint8(30)
memory usage: 7.6 MB
```

In [58]:
```
# Saving column names for future dataframe
colnames = test.columns
# Creating minmax scaler instance
mm_scaler = preprocessing.MinMaxScaler()
# Transforming data into scaled array
df_mm = mm_scaler.fit_transform(test)
# Creating new dataframe with scaled data
test_clean = pd.DataFrame(df_mm, columns=colnames)
```

In [59]:
```
# Matching column names from training dataset
train_cols = train_clean.drop(columns='y')
keepcols = train_cols.columns
test_clean = test_clean[keepcols]
```

Predicting with Random Forest Classifier

In [62]:
```
features = train_clean.drop(columns='y')

# Running Model on training data
rf = RandomForestClassifier(class_weight='balanced', max_depth=10, n_est
imators=300, n_jobs=-1, verbose=1)
rf.fit(features, train_clean['y'])
predictions = rf.predict_proba(test_clean)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   12.0s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   19.0s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 300 out of 300 | elapsed:    0.2s finished
```

In [63]:
```
# Saving 1 predictions to dataframe
rf_predictions = pd.DataFrame(predictions[:,1])
```

In [65]:
```
# Saving Random Forest predictions to file
rf_predictions.to_csv('RFPredictions_EKirkland.csv')
```

Predicting with Neural Network Classifier

In [69]:
```
features = train_clean.drop(columns='y')

# Running Model on training data
mlp = MLPClassifier(hidden_layer_sizes=(32,32), max_iter=500)
mlp.fit(features, train_clean['y'])
predictions = mlp.predict_proba(test_clean)
```

In [70]:
```
# Saving 1 predictions to dataframe
nn_predictions = pd.DataFrame(predictions[:,1])
# Saving Neural Network predictions to file
nn_predictions.to_csv('NNPredictions_EKirkland.csv')
```