Credit Card Fraud Detection Project

Imports

In [3]:

```
import numpy as np
import pandas as pd
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.model_selection import train_test_split
from sklearn import linear_model, datasets, metrics
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.calibration import CalibratedClassifierCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import log_loss
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.utils import compute_class_weight
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import BaseLogger, ModelCheckpoint, EarlyStopping, ReduceLROnPlateau,
import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')
import seaborn as sns
```

Logistic Regression and Random Forest Models

In [37]:

```
# Reading dataset
input_data = pd.read_csv("creditcard.csv")

input_data = input_data.drop(['Time', 'Amount'], axis=1) #Testing shows that with the `Tir
```

In [38]:

```
# Setting labels and features
y = input_data['Class']
X = input_data.drop(['Class'], axis=1)
```

In [39]:

```
# Shuffle and split the data into training and testing subsets
X_training, X_testing, y_training, y_testing = train_test_split(X, y, test_size = 0.4, rar
```

In [43]:

```
# Classifying with Logistic Regression
lm_classifier = LogisticRegression()
lm_classifier.fit(X,y)
y_pred = lm_classifier.predict(X_training)
lm_classifier.score(X_testing, y_testing) # Accuracy
precision = precision_score(y_training, y_pred, average='binary')
recall = recall_score(y_training, y_pred, average='binary')

cm = confusion_matrix(y_training, y_pred)

df_cm = pd.DataFrame(cm2, index = ['TP', 'TN'])
df_cm.columns = ['Predicted Pos', 'Predicted Neg']
```

```
print("\n Precision: ", precision, "\n Recall: ", recall)
sns.heatmap(df_cm, annot=True, fmt="d")
```

```
Precision:  0.868544600939
Recall:  0.631399317406
```

Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a489128>
```

In [45]:

```
# Classifying with Random Forest
rf_classifier = RandomForestClassifier()
rf_classifier = rf_classifier.fit(X_training, y_training)

y_pred = rf_classifier.predict(X_training)
rf_classifier.score(X_testing, y_testing) # Accuracy
precision = precision_score(y_training, y_pred, average='binary')
recall = recall_score(y_training, y_pred, average='binary')

cm3 = confusion_matrix(y_training, y_pred)

df_cm3 = pd.DataFrame(cm3, index = ['TP', 'TN'])
df_cm3.columns = ['Predicted Pos', 'Predicted Neg']

print("\n Precision: ", precision, "\n Recall: ", recall)
sns.heatmap(df_cm3, annot=True, fmt="d")
```

```
Precision:  0.996402877698
Recall:  0.945392491468
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a489128>
```

Neural Network approach

In [46]:

```
# Re-Importing and normalizing data with StandardScaler
input_data = pd.read_csv('creditcard.csv')
input_data.iloc[:, 1:29] = StandardScaler().fit_transform(input_data.iloc[:, 1:29])
data_matrix = input_data.as_matrix()
X = data_matrix[:, 1:29]
Y = data_matrix[:, 30]
class_weights = dict(zip([0, 1], compute_class_weight('balanced', [0, 1], Y)))
```

In [47]:

```
# k-fold cross-validation
seed = 3
np.random.seed(seed)
kfold = StratifiedKFold(n_splits=3, shuffle=True, random_state=seed)
cvscores = []
predictions = np.empty(len(Y))
predictions[:] = np.NAN
proba = np.empty([len(Y), kfold.n_splits])
proba[:] = np.NAN
k = 0
```

In [48]:

```
for train, test in kfold.split(X, Y):
    model = Sequential()
    model.add(Dense(28, input_dim=28))
    model.add(Dropout(0.2))
    model.add(Dense(22))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))
    bl = BaseLogger()
    cp = ModelCheckpoint(filepath="checkpoint.hdf5", verbose=1, save_best_only=True)
    es = EarlyStopping(monitor='val_loss', min_delta=1e-4, patience=8, verbose=0, mode='au
```

```
    rlop = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=8, min_lr=0.001)
    tb = TensorBoard(log_dir='./logs', histogram_freq=0, write_graph=True, write_images=Fa

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['binary_accuracy'
    model.fit(X[train], Y[train], batch_size=1000, nb_epoch=100, verbose=0, shuffle=True,

    # Current iteration score
    scores = model.evaluate(X[test], Y[test], verbose=1)
    print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1] * 100)
    # Storing predicted probs
    proba[train, k] = model.predict_proba(X[train]).flatten()
    k += 1
```

```
/Users/Micah/anaconda3/lib/python3.6/site-packages/keras/models.py:834: UserWarning: The `
nb_epoch` argument in `fit` has been renamed `epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '
Epoch 00000: val_loss improved from inf to 0.32502, saving model to checkpoint.hdf5
Epoch 00001: val_loss improved from 0.32502 to 0.16435, saving model to checkpoint.hdf5
Epoch 00002: val_loss improved from 0.16435 to 0.12352, saving model to checkpoint.hdf5
Epoch 00003: val_loss did not improve
Epoch 00004: val_loss improved from 0.12352 to 0.10091, saving model to checkpoint.hdf5
Epoch 00005: val_loss did not improve
Epoch 00006: val_loss improved from 0.10091 to 0.09958, saving model to checkpoint.hdf5
Epoch 00007: val_loss did not improve
Epoch 00008: val_loss did not improve
Epoch 00009: val_loss did not improve
Epoch 00010: val_loss did not improve
Epoch 00011: val_loss did not improve
Epoch 00012: val_loss did not improve
Epoch 00013: val_loss did not improve
Epoch 00014: val_loss did not improve
Epoch 00015: val_loss improved from 0.09958 to 0.09540, saving model to checkpoint.hdf5
Epoch 00016: val_loss did not improve
Epoch 00017: val_loss did not improve
Epoch 00018: val_loss did not improve
Epoch 00019: val_loss did not improve
Epoch 00020: val_loss did not improve
Epoch 00021: val_loss did not improve
Epoch 00022: val_loss did not improve
Epoch 00023: val_loss did not improve
Epoch 00024: val_loss did not improve
93856/94936 [============================>.] - ETA: 0
s□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
binary_accuracy: 97.42%
187296/189871 [============================>.] - ETA: 0s
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
poch 00000: val_loss improved from inf to 0.25271, saving model to checkpoint.hdf5
Epoch 00001: val_loss improved from 0.25271 to 0.14718, saving model to checkpoint.hdf5
Epoch 00002: val_loss improved from 0.14718 to 0.11362, saving model to checkpoint.hdf5
Epoch 00003: val_loss improved from 0.11362 to 0.10580, saving model to checkpoint.hdf5
Epoch 00004: val_loss did not improve
Epoch 00005: val_loss did not improve
Epoch 00006: val_loss did not improve
Epoch 00007: val_loss did not improve
Epoch 00008: val_loss did not improve
Epoch 00009: val_loss did not improve
Epoch 00010: val_loss did not improve
Epoch 00011: val_loss did not improve
Epoch 00012: val_loss did not improve
92800/94936 [============================>.] - ETA: 0
s□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
binary_accuracy: 97.58%
188128/189871 [============================>.] - ETA: 0s
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
poch 00000: val_loss improved from inf to 0.30139, saving model to checkpoint.hdf5
Epoch 00001: val_loss improved from 0.30139 to 0.18896, saving model to checkpoint.hdf5
Epoch 00002: val_loss improved from 0.18896 to 0.16150, saving model to checkpoint.hdf5
Epoch 00003: val_loss improved from 0.16150 to 0.14016, saving model to checkpoint.hdf5
Epoch 00004: val_loss improved from 0.14016 to 0.12446, saving model to checkpoint.hdf5
Epoch 00005: val_loss improved from 0.12446 to 0.12398, saving model to checkpoint.hdf5
Epoch 00006: val_loss improved from 0.12398 to 0.10872, saving model to checkpoint.hdf5
Epoch 00007: val_loss did not improve
Epoch 00008: val_loss did not improve
Epoch 00009: val_loss did not improve
Epoch 00010: val_loss did not improve
Epoch 00011: val_loss did not improve
Epoch 00012: val_loss did not improve
Epoch 00013: val_loss did not improve
Epoch 00014: val_loss did not improve
Epoch 00015: val_loss did not improve
93920/94935 [============================>.] - ETA: 0
s
binary_accuracy: 97.53%
188320/189872 [============================>.] - ETA: 0s
```

In [54]:

```python
pred = np.nanmean(proba, 1) > 0.5
pred = pred.astype(int)
print(classification_report(Y, pred))
# Print
pd.crosstab(Y, pred, rownames=['Actual values'], colnames=['Predictions'])
```

```
             precision    recall  f1-score   support

        0.0       1.00      0.98      0.99    284315
        1.0       0.06      0.92      0.12       492

avg / total       1.00      0.98      0.99    284807
```

Out[54]:

Predictions	0	1
Actual values		
0.0	277752	6563
1.0	40	452