test

November 2, 2017

1 Deep Learning HW1

1.1 by *Won Kim*

In [23]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns; sns.set()
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import *
from imblearn.under_sampling import *
from imblearn.combine import *
from sklearn.metrics import *
```

In [24]:
```
df = pd.read_csv("train.csv")
df.tail()
```

Out[24]:
```
           Time        V1        V2        V3        V4        V5        V6  \
29195  172762.0 -0.725459  0.194981 -1.785571 -3.779860  2.177420  2.975713
29196  172764.0 -0.764523  0.588379 -0.907599 -0.418847  0.901528 -0.760802
29197  172767.0 -0.268061  2.540315 -1.400915  4.846661  0.639105  0.186479
29198  172785.0  0.120316  0.931005 -0.546012 -0.745097  1.130314 -0.235973
29199  172788.0  1.919565 -0.301254 -3.249640 -0.557828  2.630515  3.031260

             V7        V8        V9  ...        V21       V22       V23  \
29195 -0.239695  0.912303 -3.159994  ...   0.046103  0.102480 -0.461027
29196  0.758545  0.414698 -0.730854  ...   0.003530 -0.431876  0.141759
29197 -0.045911  0.936448 -2.419986  ...  -0.263889 -0.857904  0.235172
29198  0.812722  0.115093 -0.204064  ...  -0.314205 -0.808520  0.050343
29199 -0.296827  0.708417  0.432454  ...   0.232045  0.578229 -0.037501

            V24       V25       V26       V27       V28  Amount  Class
29195  0.717647  1.253036  0.207138 -0.630549 -0.163911    7.00      0
29196  0.587119 -0.200998  0.267337 -0.152951 -0.065285   80.00      0
29197 -0.681794 -0.668894  0.044657 -0.066751 -0.072447   12.82      0
29198  0.102800 -0.435870  0.124079  0.217940  0.068803    2.69      0
29199  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88      0
```

```
[5 rows x 31 columns]
```

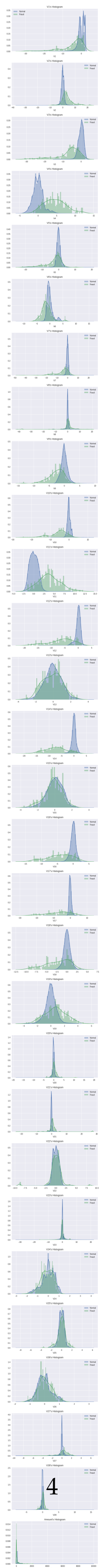
In [25]:
```
df.isnull().sum()
```

Out[25]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

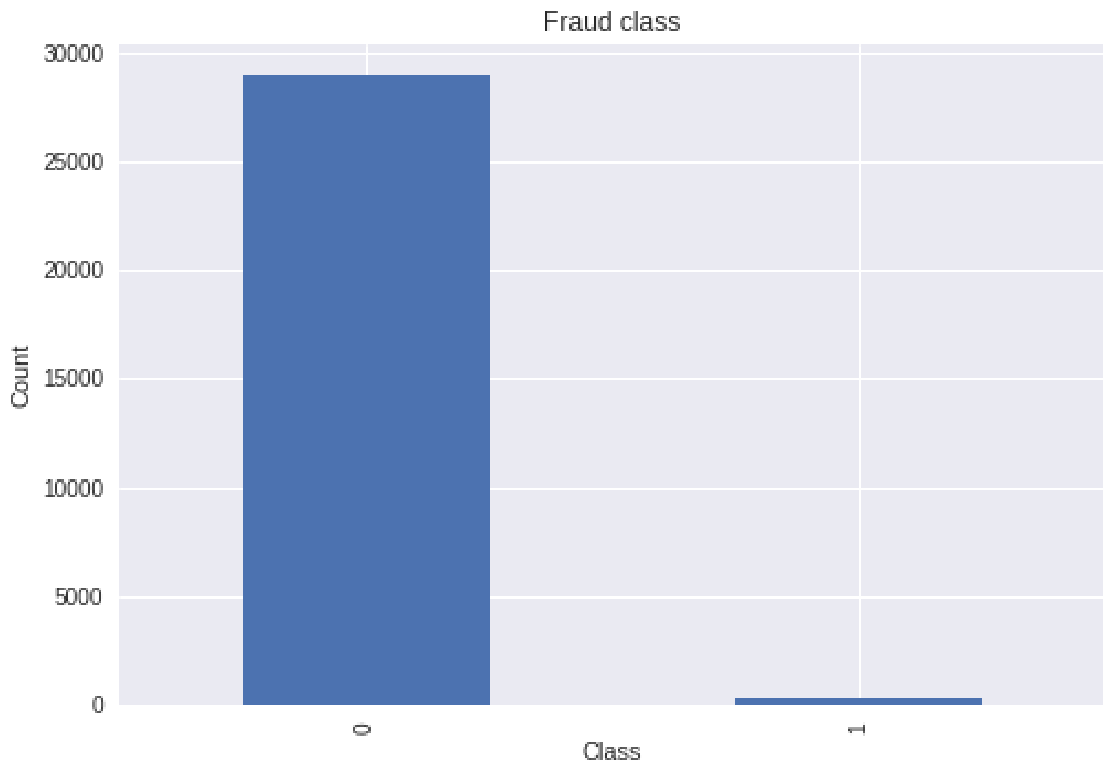
In [26]:
```
columns = list(df.columns)[1:30] # except time, class
```

In [50]:
```
plt.figure(figsize = (7, 4*28))
for idx, column in enumerate(columns):
    plt.subplot(29, 1, idx+1)
    sns.distplot(df[df["Class"] == 0][column], bins = 50, label = "Normal")
    sns.distplot(df[df["Class"] == 1][column], bins = 50, label = "Fraud")
    plt.title(str(column) + "'s Histogram")
    plt.legend()
```

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



```
In [27]: count_classes = pd.value_counts(df.Class)
         count_classes.plot(kind = 'bar')
         plt.title("Fraud class")
         plt.xlabel("Class")
         plt.ylabel("Count")
         plt.show()
         print("The total counts of each class are ", count_classes[0], "for class 0 and ", coun
```




```
The total counts of each class are  28908 for class 0 and  292  for class 1
```

1.1.1 Train set preprocessing

```
In [42]: def preprocessing(df):
             df = df.drop(['V28','V27','V26','V25','V24','V23','V22','V20','V15','V13','V8'], ax
             df['scaledAmount'] = (df['Amount'] - df['Amount'].mean()) / df['Amount'].std()
             scaled_df = df.copy()
             preprocessed_df = scaled_df.drop(['Time', 'Amount'], axis = 1)
             return preprocessed_df

         preprocessed_df = preprocessing(df)
```

```
In [43]: X_train, y_train = preprocessed_df.drop("Class", axis = 1), preprocessed_df.Class
```

1.1.2 Validation set preprocessing

```
In [44]: val_df = pd.read_csv("valid.csv")
         preprocessed_val_df = preprocessing(val_df)
```

```
In [45]: X_val, y_val = preprocessed_val_df.drop("Class", axis = 1), preprocessed_val_df.Class
```

1.1.3 Model Learning 1. Random Forest with SMOTE Resampling

```
In [46]: sm = SMOTE(random_state=0)
         X_train_res, y_train_res = sm.fit_sample(X_train, y_train)

         n_estimators = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150] # 100
         max_depths = list(range(5, 20))
```

```
In [47]: def RFClassifier_and_score(n_estimator, max_depth):
             clf_rf = RandomForestClassifier(n_estimators=n_estimator, max_depth = max_depth, ra
             clf_rf.fit(X_train_res, y_train_res)
             predict = clf_rf.predict(X_val)
             print('\nResults with {} estimators and {} max_depth'.format(n_estimator, max_depth
             #print("Train")
             #print("recall: ", recall_score(y_train_res, predict))
             #print("precision: ", precision_score(y_train_res, predict))
             #print("f1 measure: ", f1_score(y_train_res, predict))

             print("\nValidation")
             print("recall: ", recall_score(y_val, predict))
             print("precision: ", precision_score(y_val, predict))
             print("f1 measure: ", f1_score(y_val, predict))
```

```
In [49]: for n_estimator in n_estimators:
             RFClassifier_and_score(n_estimator, max_depth = None)
```

```
Results with 10 estimators and None max_depth

Validation
recall:  0.81
precision:  0.870967741935
f1 measure:  0.839378238342

Results with 20 estimators and None max_depth

Validation
recall:  0.81
precision:  0.910112359551
f1 measure:  0.857142857143
```

```
Results with 30 estimators and None max_depth

Validation
recall:  0.81
precision:  0.920454545455
f1 measure:  0.86170212766

Results with 40 estimators and None max_depth

Validation
recall:  0.81
precision:  0.920454545455
f1 measure:  0.86170212766

Results with 50 estimators and None max_depth

Validation
recall:  0.81
precision:  0.920454545455
f1 measure:  0.86170212766

Results with 60 estimators and None max_depth

Validation
recall:  0.81
precision:  0.920454545455
f1 measure:  0.86170212766

Results with 70 estimators and None max_depth

Validation
recall:  0.81
precision:  0.941860465116
f1 measure:  0.870967741935

Results with 80 estimators and None max_depth

Validation
recall:  0.81
precision:  0.931034482759
f1 measure:  0.866310160428

Results with 90 estimators and None max_depth

Validation
recall:  0.81
precision:  0.931034482759
```

```
f1 measure:  0.866310160428

Results with 100 estimators and None max_depth

Validation
recall:  0.81
precision:  0.941860465116
f1 measure:  0.870967741935

Results with 110 estimators and None max_depth

Validation
recall:  0.81
precision:  0.931034482759
f1 measure:  0.866310160428

Results with 120 estimators and None max_depth

Validation
recall:  0.81
precision:  0.941860465116
f1 measure:  0.870967741935

Results with 130 estimators and None max_depth

Validation
recall:  0.81
precision:  0.931034482759
f1 measure:  0.866310160428

Results with 140 estimators and None max_depth

Validation
recall:  0.81
precision:  0.920454545455
f1 measure:  0.86170212766

Results with 150 estimators and None max_depth

Validation
recall:  0.81
precision:  0.920454545455
f1 measure:  0.86170212766
```

```
In [48]: RFClassifier_and_score(100, max_depth = None)
```

```
Results with 100 estimators and None max_depth
```

```
Validation
recall:  0.81
precision:  0.941860465116
f1 measure:  0.870967741935
```

```
In [20]: for max_depth in max_depths:
             RFClassifier_and_score(100, max_depth)
```

```
Results with 100 estimators and 5 max_depth

Validation
recall:  0.83
precision:  0.691666666667
f1 measure:  0.754545454545

Results with 100 estimators and 6 max_depth

Validation
recall:  0.83
precision:  0.721739130435
f1 measure:  0.772093023256

Results with 100 estimators and 7 max_depth

Validation
recall:  0.83
precision:  0.747747747748
f1 measure:  0.78672985782

Results with 100 estimators and 8 max_depth

Validation
recall:  0.83
precision:  0.775700934579
f1 measure:  0.80193236715

Results with 100 estimators and 9 max_depth

Validation
recall:  0.83
precision:  0.805825242718
f1 measure:  0.817733990148

Results with 100 estimators and 10 max_depth
```

```
Validation
recall:  0.82
precision:  0.803921568627
f1 measure:  0.811881188119

Results with 100 estimators and 11 max_depth

Validation
recall:  0.82
precision:  0.836734693878
f1 measure:  0.828282828283

Results with 100 estimators and 12 max_depth

Validation
recall:  0.82
precision:  0.854166666667
f1 measure:  0.836734693878

Results with 100 estimators and 13 max_depth

Validation
recall:  0.82
precision:  0.881720430108
f1 measure:  0.849740932642

Results with 100 estimators and 14 max_depth

Validation
recall:  0.82
precision:  0.911111111111
f1 measure:  0.863157894737

Results with 100 estimators and 15 max_depth

Validation
recall:  0.82
precision:  0.863157894737
f1 measure:  0.841025641026

Results with 100 estimators and 16 max_depth

Validation
recall:  0.82
precision:  0.901098901099
f1 measure:  0.858638743455

Results with 100 estimators and 17 max_depth
```

```
Validation
recall:  0.82
precision:  0.911111111111
f1 measure:  0.863157894737

Results with 100 estimators and 18 max_depth

Validation
recall:  0.81
precision:  0.910112359551
f1 measure:  0.857142857143

Results with 100 estimators and 19 max_depth

Validation
recall:  0.81
precision:  0.920454545455
f1 measure:  0.86170212766
```

To avoid over-fitting in random forest, the main thing you need to do is optimize a tuning parameter that governs the number of features that are randomly chosen to grow each tree from the bootstrapped data. Typically, you do this via k-fold cross-validation, where k{5,10}, and choose the tuning parameter that minimizes test sample prediction error. In addition, growing a larger forest will improve predictive accuracy, although there are usually diminishing returns once you get up to several hundreds of trees.

In [113]:
```python
val_predicted_labels = clf_rf.predict(X_val)
new_column = pd.DataFrame({'Predicted Class': val_predicted_labels})
val_df["Predicted Class"] = new_column
```

In [110]:
```python
val_df.to_csv('valid_check.csv', index=False)
```

2 Test

In [67]:
```python
test_df = pd.read_csv("test.csv")
preprocessed_test_df = preprocessing(test_df)
```

In [68]:
```python
selected_n_estimators = 100
clf_rf = RandomForestClassifier(n_estimators=selected_n_estimators, random_state=0)
clf_rf.fit(X_train_res, y_train_res)
predicted_labels = clf_rf.predict(preprocessed_test_df)
```

2.0.1 Write to result.csv

In [97]:
```python
result_df = pd.read_csv('result.csv')
new_column = pd.DataFrame({'Class': predicted_labels})
```

```
        result_df = result_df.drop("Class", axis = 1)
        result_df["Class"] = new_column
```

```
In [100]: result_df.to_csv('result.csv', index=False)
```

2.1 # Another approach

2.1.1 Model Learning 2. Gaussian Mixture Model

```
In [70]: from sklearn.mixture import GMM
```

```
In [71]: gmm = GMM(n_components=2, random_state=0)
         labels = gmm.fit(X_train_res).predict(X_val)
         set(labels)
```

```
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:57: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
```

```
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
/home/snu/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py:75: DeprecationWarn
  warnings.warn(msg, category=DeprecationWarning)
```

```
Out[71]: {0, 1}
```

```
In [72]: print("recall: ", recall_score(y_val, labels))
         print("precision: ", precision_score(y_val, labels))
         print("f1 measure: ", f1_score(y_val, labels))
```

```
recall:  0.81
precision:  0.20822622108
f1 measure:  0.331288343558
```

Gaussian Mixture Model gives me worse result.