Credit Card Fraud Detection

Chua Yeow Long

Abstract

Credit card fraud detection dataset is used for the final project. We'll explore the use of various machine learning algorithms to see if we get detect fraudulent credit card transactions. How do we handle imbalanced datasets? Do we simply just ignore them? What is the best result we can achieve using state-of-the-art methods and techniques. Machine learning algorithms used include neural networks, logistic regression, XGBoost, random forest and support vector machine. We will also take a look at how to handle imbalanced datasets such as using undersampling using near miss and oversampling using SMOTE. We see that Support Vector Machine coupled with SMOTE oversampling we are able to achieve 0 false negatives and a high accuracy score of 0.9993.

Motivation

Identify fraudulent credit card transactions.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Dataset(s)

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation.

Data Preparation and Cleaning

We need to remove the null and na values from the dataset as these entries will cause errors in our modelling of the data.

As we do not have a designated test data to evaluate the performance of our machine learning models, a train test split is performed and K-Fold cross validation can be used to better tune hyper parameters of our machine learning models.

Even though the dataset is PCA’ed which is assumed to be normalized, we normalize the data using scikit learn’s MinMaxScaler before feeding the data into our machine learning models so as not to introduce unnecessary bias into our data.

Research Question(s)

Explore the use of various machine learning algorithms to see if we get detect fraudulent credit card transactions.

How do we handle imbalanced datasets? Do we simply just ignore them?

What is the best result we can achieve using state-of-the-art methods and techniques.

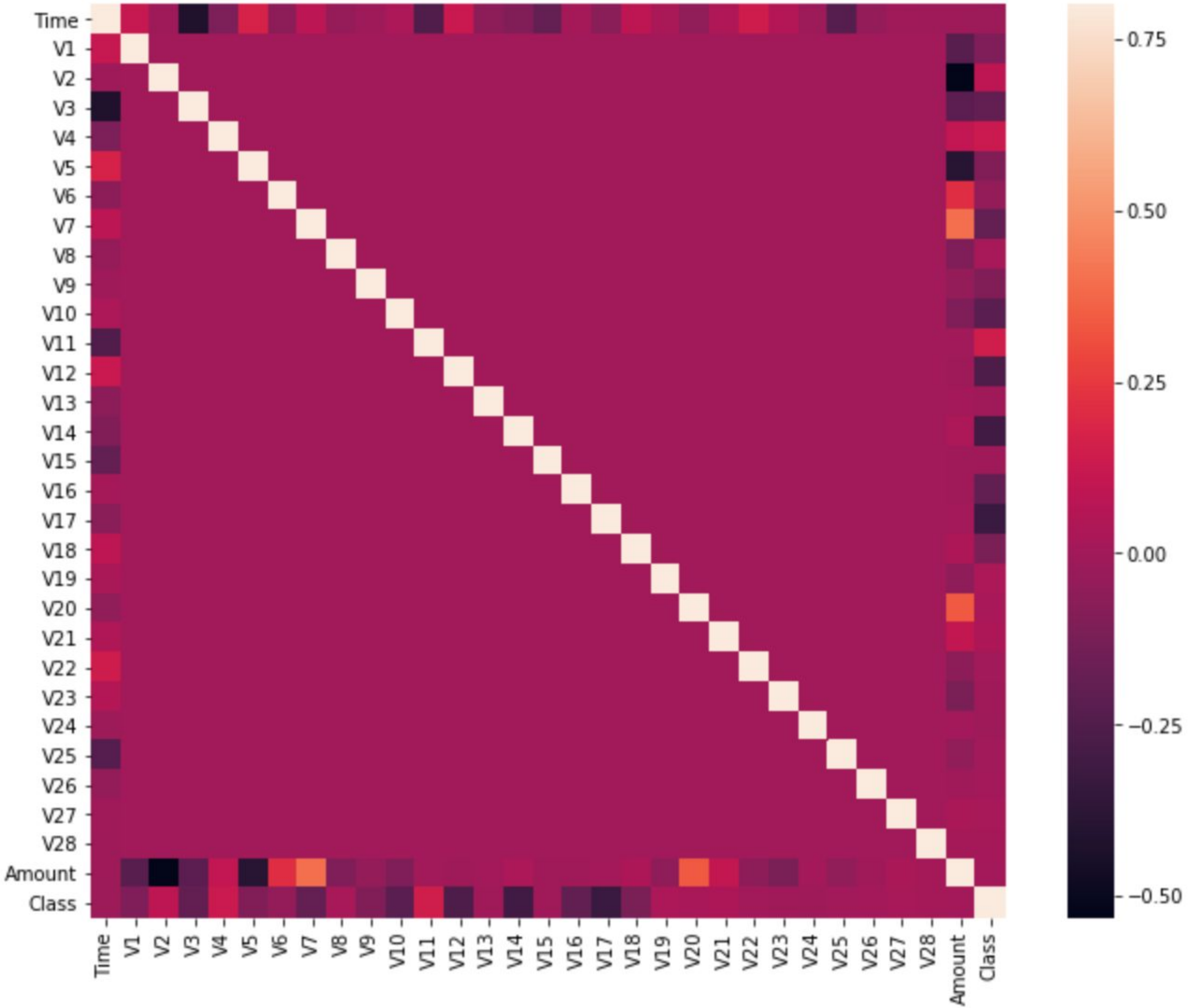
Methods

To analyze the data, I have used the following machine learning algorithms. They are a simple neural network, logistic regression, XGBoost, random forest and support vector machine.

The dataset is imbalanced and in the notebook, i have explored the use of two methods to handle this. They are undersampling (near miss) and oversampling (SMOTE). From the analysis, we see that oversampling using SMOTE which greatly improves the precision and recall of the machine learning model. We are able to achieve a perfect zero false negative and this is great!

Findings

We see that the features of the dataset is largely uncorrelated. This also means that the data is not sparse. This features are good for machine learning.



Findings

We see that by omitting the preprocessing step of oversampling and undersampling, the performance of our machine learning models are reduced. We explored the use of various machine learning algorithms such as neural networks, logistic regression, XGBoots, random forest and support vector machine and we see that an accuracy score of 0.9993 is achieved using support vector machine and oversampling using smote.

Limitations

We achieved a near zero false negative and a limited number of false positive with this dataset. In this case, false positive is not really an issue as it is better to be safe than sorry. False negatives is the parameter we want to keep track of. It is not always the case that we can achieve a zero false negative.

Also, the dataset consists of results after a PCA transformation, it makes me wonder if the V1-V28 variables are necessary to analyze this dataset. Are using the time and amount variables enough when analyzing this data?

Conclusions

We have explored the use of various machine learning algorithms such as neural networks, logistic regression, XGBoost, random forest and support vector machines.

The imbalanced dataset, if left unhandled, will decrease the performance of our machine learning models. We have explored the use of undersampling using near miss and oversampling using smote to address this.

Support Vector Machine coupled with over sampling using smote we are able to achieve an accuracy score of 0.9993 and 0 false negatives and limited number of false positives.

Acknowledgements

As the dataset is hosted on Kaggle, I would like to thank for entire kaggle community for the knowledge sharing and resource.

References

This is a dataset hosted on Kaggle.

https://www.kaggle.com/mlg-ulb/creditcardfraud

In [0]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from dateutil import parser
%matplotlib inline


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report,
accuracy_score
from sklearn.model_selection import GridSearchCV
import pickle
from lightgbm import LGBMClassifier


#deep learning libraries
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
```

```
Using TensorFlow backend.
```

In [0]:

```
data = pd.read_csv('creditcard.csv')
data.shape
```

Out[0]:

```
(284807, 31)
```

In [0]:

```
data.head()
```

Out[0]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270

5 rows × 31 columns

In [0]:
```
#class imbalance

data.Class.value_counts()
```

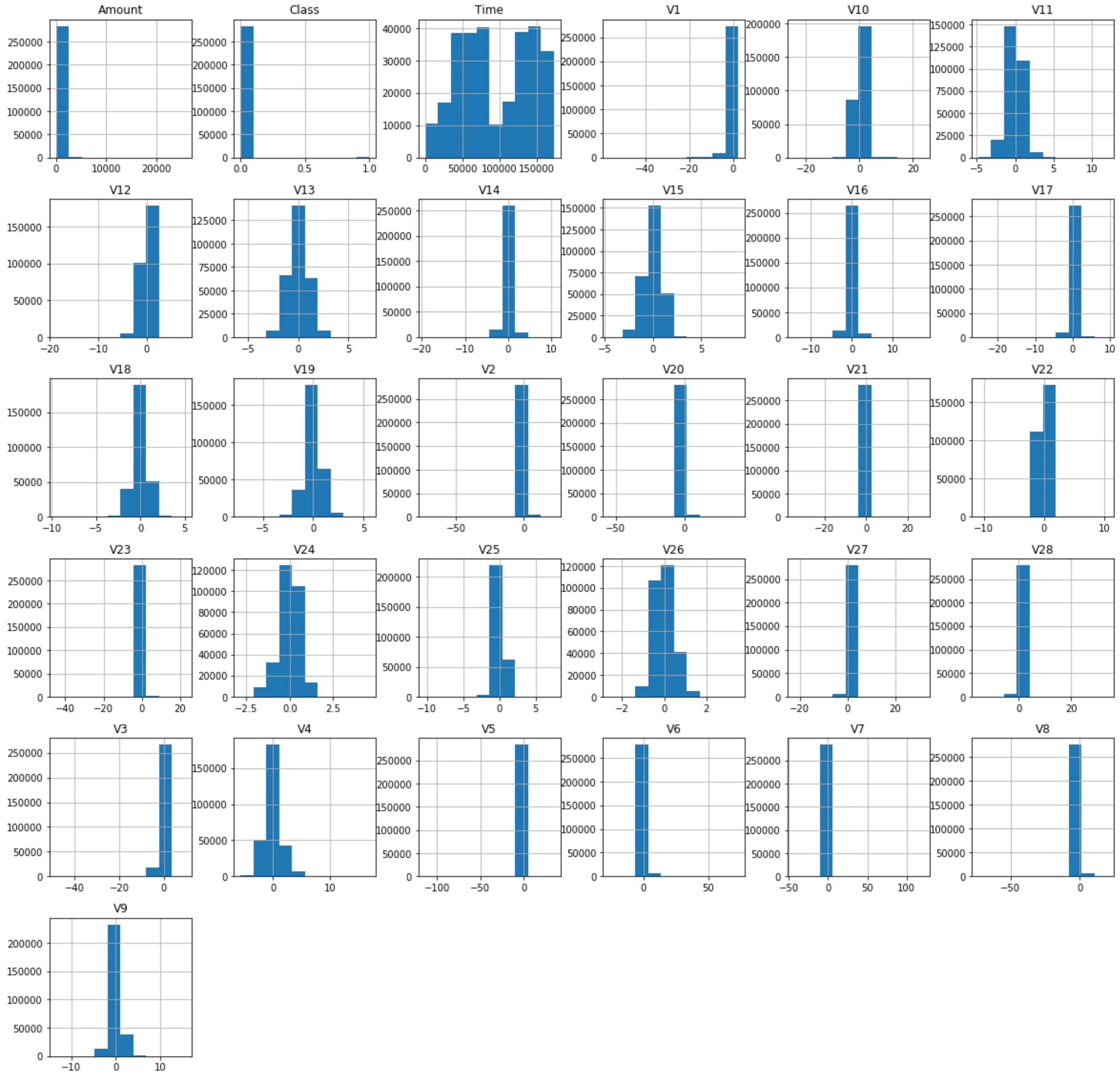
Out[0]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [0]:
```
X = data.drop(labels='Class',axis=1)
Y = data['Class']
```

In [0]:
```
type(X)
```

Out[0]:
```
pandas.core.frame.DataFrame
```

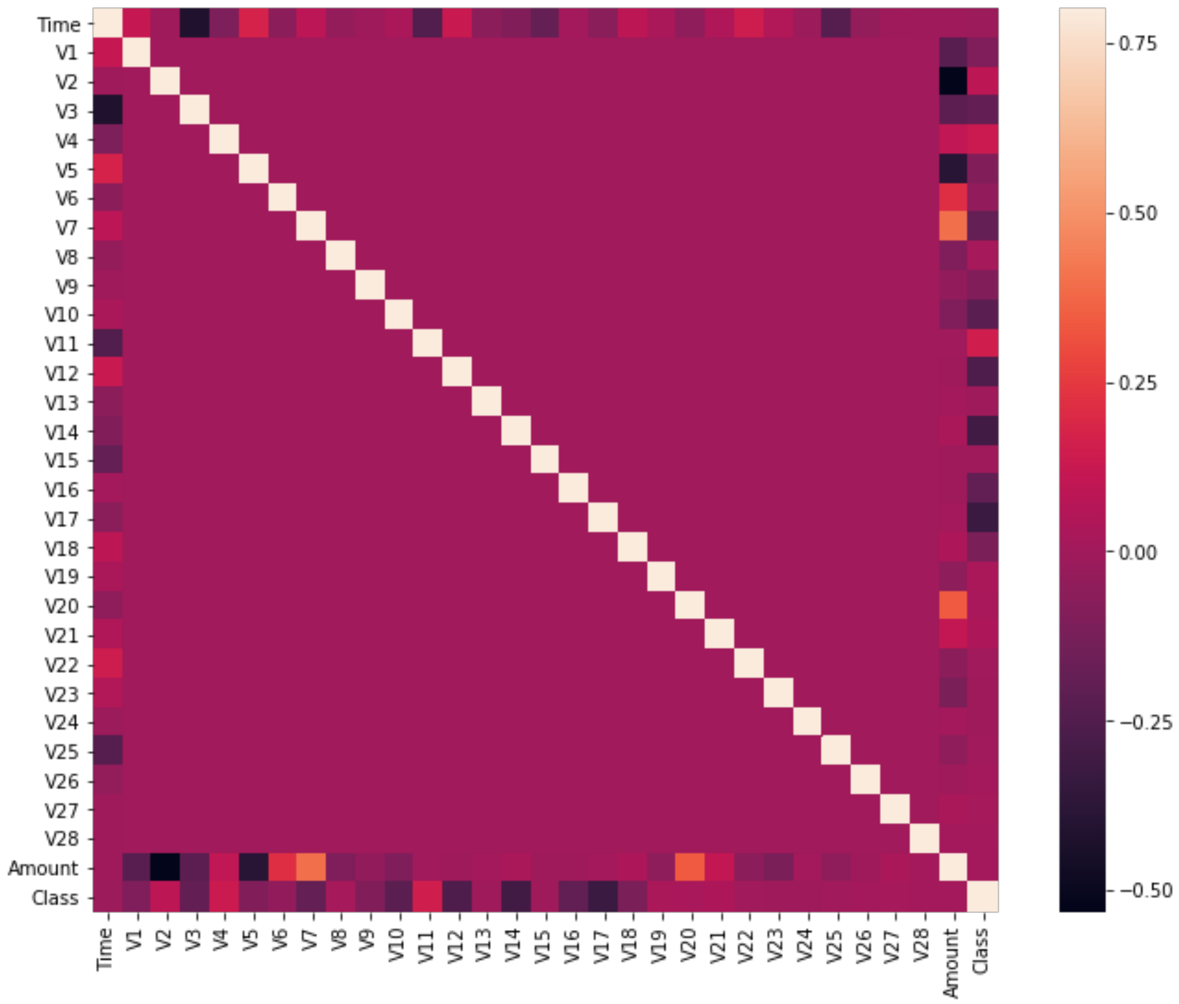
In [0]:
```
data.hist(figsize = (20, 20))
plt.show()
```



In [0]:

```
# Correlation matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



Feature Engineering

In [0]:

```
SS = StandardScaler()
X['normAmount'] = SS.fit_transform(X['Amount'].values.reshape(-1, 1))
X = X.drop(['Time','Amount'],axis=1)
X.head()
```

Out[0]:

	V1	V2	V3	V4	V5	V6	V7	V8	
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0

5 rows × 29 columns

In [0]:

```
np.random.seed(10)
x_train,x_test,y_train,y_test = train_test_split(X,Y, test_size = 0.2)
```

In [0]:

```
x_train.shape,x_test.shape
```

Out[0]:

```
((227845, 29), (56962, 29))
```

Model Fitting

Simple Neural Network

In [0]:

```
model = Sequential([
    Dense(units=16,input_dim = 29, activation = 'relu'),
    Dense(units=24, activation = 'relu'),
    Dropout(0.5),
    Dense(units=20, activation = 'relu'),
    Dense(units=24, activation = 'relu'),
    Dense(units=1, activation = 'sigmoid'),
])
```

```
WARNING:tensorflow:From C:\Users\user\Anaconda3\lib\site-packages\
tensorflow\python\framework\op_def_library.py:263: colocate_with (
from tensorflow.python.framework.ops) is deprecated and will be re
moved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\user\Anaconda3\lib\site-packages\
keras\backend\tensorflow_backend.py:3445: calling dropout (from te
nsorflow.python.ops.nn_ops) with keep_prob is deprecated and will
be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `r
ate = 1 - keep_prob`.
```

In [0]:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 16)                480
_________________________________________________________________
dense_2 (Dense)              (None, 24)                408
_________________________________________________________________
dropout_1 (Dropout)          (None, 24)                0
_________________________________________________________________
dense_3 (Dense)              (None, 20)                500
_________________________________________________________________
dense_4 (Dense)              (None, 24)                504
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 25
=================================================================
Total params: 1,917
Trainable params: 1,917
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=[
'accuracy'])
model.fit(x_train,y_train,batch_size=15,epochs=5)
```

```
WARNING:tensorflow:From C:\Users\user\Anaconda3\lib\site-packages\
tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.
python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/5
227845/227845 [==============================] - 13s 56us/step - l
oss: 0.0086 - acc: 0.9985
Epoch 2/5
227845/227845 [==============================] - 12s 53us/step - l
oss: 0.0041 - acc: 0.9993
Epoch 3/5
227845/227845 [==============================] - 12s 53us/step - l
oss: 0.0038 - acc: 0.9993
Epoch 4/5
227845/227845 [==============================] - 12s 55us/step - l
oss: 0.0036 - acc: 0.9993
Epoch 5/5
227845/227845 [==============================] - 13s 55us/step - l
oss: 0.0034 - acc: 0.9994
```

Out[0]:

```
<keras.callbacks.History at 0x1f489b2e0b8>
```

In [0]:

```
print(model.evaluate(x_test,y_test))
```

```
56962/56962 [==============================] - 1s 9us/step
[0.0032044619671679914, 0.9995259997893332]
```

In [0]:

```
# Functionalize model fittting

def FitModel(X,Y,algo_name,algorithm,gridSearchParams,cv):
    np.random.seed(10)
    x_train,x_test,y_train,y_test = train_test_split(X,Y, test_size
= 0.2)


    grid = GridSearchCV(
        estimator=algorithm,
        param_grid=gridSearchParams,
        cv=cv, scoring='accuracy', verbose=1, n_jobs=-1)


    grid_result = grid.fit(x_train, y_train)
    best_params = grid_result.best_params_
    pred = grid_result.predict(x_test)
    cm = confusion_matrix(y_test, pred)
   # metrics =grid_result.gr
    print(pred)
    #pickle.dump(grid_result,open(algo_name,'wb'))

    print('Best Params :',best_params)
    print('Classification Report :',classification_report(y_test,pr
ed))
    print('Accuracy Score : ' + str(accuracy_score(y_test,pred)))
    print('Confusion Matrix : \n', cm)
```

Logistic Regression

```
In [0]: # Create regularization penalty space
        penalty = ['l1', 'l2']

        # Create regularization hyperparameter space
        C = np.logspace(0, 4, 10)

        # Create hyperparameter options
        hyperparameters = dict(C=C, penalty=penalty)

        FitModel(X,Y,'LogisticRegression_norm',LogisticRegression(),hyperpa
        rameters,cv=5)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 39.4min fini
shed
C:\Users\user\Anaconda3\lib\site-packages\sklearn\linear_model\log
istic.py:432: FutureWarning: Default solver will be changed to 'lb
fgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

[0 0 0 ... 0 0 0]
Best Params : {'C': 166.81005372000593, 'penalty': 'l1'}
Classification Report :               precision    recall  f1-scor
e   support

           0       1.00      1.00      1.00     56868
           1       0.88      0.67      0.76        94

    accuracy                           1.00     56962
   macro avg       0.94      0.84      0.88     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy Score : 0.9992977774656788
Confusion Matrix :
 [[56859     9]
 [   31    63]]
```

XgBoost

In [0]:

```
param ={
            'n_estimators': [100, 500, 1000, 2000],

        }
FitModel(X,Y,'XGBoost_norm',XGBClassifier(),param,cv=5)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  18 out of  20 | elapsed: 20.5min rema
ining:  2.3min
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed: 20.6min fini
shed

[0 0 0 ... 0 0 0]
Best Params : {'n_estimators': 1000}
Classification Report :                precision    recall  f1-scor
e   support

           0       1.00      1.00      1.00     56868
           1       0.98      0.84      0.90        94

    accuracy                           1.00     56962
   macro avg       0.99      0.92      0.95     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy Score : 0.9997015554229135
Confusion Matrix :
 [[56866     2]
 [   15    79]]
```

Random Forest

In [0]:

```
param ={
            'n_estimators': [100, 500, 1000, 2000],

        }
FitModel(X,Y,'Random Forest',RandomForestClassifier(),param,cv=5)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  18 out of  20 | elapsed: 55.5min rema
ining:  6.2min
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed: 57.7min fini
shed

[0 0 0 ... 0 0 0]
Best Params : {'n_estimators': 2000}
Classification Report :               precision    recall  f1-scor
e   support

           0       1.00      1.00      1.00     56868
           1       0.99      0.82      0.90        94

    accuracy                           1.00     56962
   macro avg       0.99      0.91      0.95     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy Score : 0.9996839998595555
Confusion Matrix :
 [[56867     1]
 [   17    77]]
```

SVC

In [0]:

```
param ={
            'C': [0.1, 1, 100, 1000],
            'gamma': [0.0001, 0.001, 0.005, 0.1, 1, 3, 5]
        }
FitModel(X,Y,'SVC_norm',SVC(),param,cv=5)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed: 33.6min
[Parallel(n_jobs=-1)]: Done 140 out of 140 | elapsed: 593.5min fin
ished

[0 0 0 ... 0 0 0]
Best Params : {'C': 100, 'gamma': 0.005}
Classification Report :               precision    recall  f1-scor
e   support

           0       1.00      1.00      1.00     56868
           1       0.96      0.82      0.89        94

    accuracy                           1.00     56962
   macro avg       0.98      0.91      0.94     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy Score : 0.9996488887328394
Confusion Matrix :
 [[56865     3]
 [   17    77]]
```

Balancing the Dataset

Under Sampling

In [0]:

```
from imblearn.under_sampling import NearMiss
```

In [0]:

```
sm =NearMiss(version=2,random_state=42)
X_res , Y_res = sm.fit_resample(X,Y)
pd.Series(Y_res).value_counts()
```

Out[0]:

```
1    492
0    492
dtype: int64
```

In [0]:

```
X_res.shape,X.shape
```

Out[0]:

```
((984, 29), (284807, 29))
```

Logistics Regression

In [0]:

```
# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter space
C = np.logspace(0, 4, 10)

# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)

FitModel(X_res,Y_res,'LogisticRegression_US',LogisticRegression(),h
yperparameters,cv=5)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done  77 out of 100 | elapsed:    1.5s rema
ining:    0.4s
```

```
[0 0 1 0 0 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 1
1 1 1 1
 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0
0 0 0 0
 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0
1 1 1 0
 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 1 1
1 1 1 1
 1 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0
1 1 1 1
 1 1 1 1 0 1 1 0 1 1 1 0]
Best Params : {'C': 7.742636826811269, 'penalty': 'l1'}
Classification Report :               precision    recall  f1-scor
e   support

           0       0.95      0.94      0.95       102
           1       0.94      0.95      0.94        95

    accuracy                           0.94       197
   macro avg       0.94      0.94      0.94       197
weighted avg       0.94      0.94      0.94       197

Accuracy Score : 0.9441624365482234
Confusion Matrix :
 [[96  6]
 [ 5 90]]

[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    1.6s fini
shed
C:\Users\user\Anaconda3\lib\site-packages\sklearn\linear_model\log
istic.py:432: FutureWarning: Default solver will be changed to 'lb
fgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

XGBOOST

In [0]:
```
param ={
            'n_estimators': [100, 500, 1000, 2000],

        }
FitModel(X_res,Y_res,'XGBoost_US',XGBClassifier(),param,cv=5)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  18 out of  20 | elapsed:    5.6s rema
ining:    0.5s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    5.7s fini
shed

[1 0 1 0 0 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 1
1 1 1 0
 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0
0 1 0 1
 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1
1 1 1 0
 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 1
1 1 1 1
 1 0 0 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0
1 1 1 1
 1 1 1 1 0 1 1 0 1 1 1 0]
Best Params : {'n_estimators': 100}
Classification Report :               precision    recall  f1-scor
e   support

           0       1.00      0.96      0.98       102
           1       0.96      1.00      0.98        95

    accuracy                           0.98       197
   macro avg       0.98      0.98      0.98       197
weighted avg       0.98      0.98      0.98       197

Accuracy Score : 0.9796954314720813
Confusion Matrix :
 [[98  4]
 [ 0 95]]
```

Random Forest

In [0]:

```
param ={
            'n_estimators': [100, 500, 1000, 2000],

        }
FitModel(X_res,Y_res,'Random Forest_US',RandomForestClassifier(),pa
ram,cv=5)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  18 out of  20 | elapsed:    3.3s rema
ining:    0.3s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    3.4s fini
shed

[0 0 1 0 0 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 1
1 1 1 0
 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0
0 1 0 0
 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1
1 1 1 0
 0 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 1
1 1 1 1
 1 0 0 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0
1 1 1 1
 1 1 1 1 0 1 1 0 1 1 1 0]
Best Params : {'n_estimators': 100}
Classification Report :               precision    recall  f1-scor
e   support

           0       0.99      0.98      0.99       102
           1       0.98      0.99      0.98        95

    accuracy                           0.98       197
   macro avg       0.98      0.98      0.98       197
weighted avg       0.98      0.98      0.98       197

Accuracy Score : 0.9847715736040609
Confusion Matrix :
 [[100   2]
 [  1  94]]
```

In [0]:

```
```

Neural Network

In [0]:

```
np.random.seed(10)
x_train,x_test,y_train,y_test = train_test_split(X_res,Y_res, test_
size = 0.2)
x_train.shape
```

Out[0]:

```
(787, 29)
```

In [0]:

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=[
'accuracy'])
model.fit(x_train,y_train,batch_size=15,epochs=5)
```

```
WARNING:tensorflow:From C:\Users\user\Anaconda3\lib\site-packages\
tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.
python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/5
787/787 [==============================] - 1s 722us/step - loss: 0
.6199 - acc: 0.6938
Epoch 2/5
787/787 [==============================] - 0s 53us/step - loss: 0.
4367 - acc: 0.8501
Epoch 3/5
787/787 [==============================] - 0s 54us/step - loss: 0.
3657 - acc: 0.8895
Epoch 4/5
787/787 [==============================] - 0s 56us/step - loss: 0.
2936 - acc: 0.9161
Epoch 5/5
787/787 [==============================] - 0s 56us/step - loss: 0.
2581 - acc: 0.9263
```

Out[0]:

```
<keras.callbacks.History at 0x201bddb7a90>
```

In [0]:

```
print(model.evaluate(x_test,y_test))
```

```
197/197 [==============================] - 0s 354us/step
[0.17325389597016544, 0.9543147211147444]
```

Over Sampling

In [0]:

```
from imblearn.over_sampling import SMOTE
```

In [0]:

```
sm =SMOTE(random_state=42)
X_res_OS , Y_res_OS = sm.fit_resample(X,Y)
```

In [0]:
```
pd.Series(Y_res_OS).value_counts()
```

Out[0]:
```
1    284315
0    284315
dtype: int64
```

Neural Network

In [0]:
```
np.random.seed(10)
x_train,x_test,y_train,y_test = train_test_split(X_res_OS,Y_res_OS,
test_size = 0.2)
x_train.shape
```

Out[0]:
```
(454904, 29)
```

In [0]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=[
'accuracy'])
model.fit(x_train,y_train,batch_size=15,epochs=5,validation_data=[x
_test,y_test])
```

```
WARNING:tensorflow:From C:\Users\user\Anaconda3\lib\site-packages\
tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.
python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Train on 454904 samples, validate on 113726 samples
Epoch 1/5
454904/454904 [==============================] - 30s 67us/step - l
oss: 0.0442 - acc: 0.9838 - val_loss: 0.0163 - val_acc: 0.9954
Epoch 2/5
454904/454904 [==============================] - 30s 65us/step - l
oss: 0.0177 - acc: 0.9949 - val_loss: 0.0106 - val_acc: 0.9977
Epoch 3/5
454904/454904 [==============================] - 30s 65us/step - l
oss: 0.0129 - acc: 0.9967 - val_loss: 0.0083 - val_acc: 0.9980
Epoch 4/5
454904/454904 [==============================] - 30s 65us/step - l
oss: 0.0115 - acc: 0.9971 - val_loss: 0.0073 - val_acc: 0.9985
Epoch 5/5
454904/454904 [==============================] - 30s 65us/step - l
oss: 0.0100 - acc: 0.9975 - val_loss: 0.0072 - val_acc: 0.9984
```

Out[0]:
```
<keras.callbacks.History at 0x1a0884bf198>
```

In [0]:
```
print(model.evaluate(x_test,y_test))
```

```
113726/113726 [==============================] - 1s 9us/step
[0.007239821667997645, 0.9983556970261858]
```

In [0]:

```
```

Logistics Regression

In [0]:

```
# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter space
C = np.logspace(0, 4, 10)

# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)

FitModel(X_res_OS,Y_res_OS,'LogisticRegression_OS',LogisticRegressi
on(),hyperparameters,cv=5)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  5.0min fini
shed
C:\Users\user\Anaconda3\lib\site-packages\sklearn\linear_model\log
istic.py:432: FutureWarning: Default solver will be changed to 'lb
fgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

[1 1 0 ... 0 1 1]
Best Params : {'C': 2.7825594022071245, 'penalty': 'l2'}
Classification Report :               precision    recall  f1-scor
e   support

           0       0.92      0.98      0.95     56989
           1       0.97      0.92      0.95     56737

    accuracy                           0.95    113726
   macro avg       0.95      0.95      0.95    113726
weighted avg       0.95      0.95      0.95    113726

Accuracy Score : 0.9468898932522026
Confusion Matrix :
 [[55637  1352]
 [ 4688 52049]]
```

SVC

In [0]:

```
param ={
            'C': [0.1, 1, 100, 1000],
            'gamma': [0.0001, 0.001, 0.005]
        }
FitModel(X_res_OS,Y_res_OS,'SVC_norm',SVC(),param,cv=5)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurren
t workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed: 206.7min
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed: 321.4min fin
ished

[1 1 0 ... 0 1 1]
Best Params : {'C': 1000, 'gamma': 0.005}
Classification Report :                precision    recall  f1-scor
e   support

           0       1.00      1.00      1.00     56989
           1       1.00      1.00      1.00     56737

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726

Accuracy Score : 0.9993053479415437
Confusion Matrix :
 [[56910    79]
 [    0 56737]]
```

In [0]:

In [0]:

In [0]:

In [0]: