Credit Card Fraud Detection

By: Garima

Date: 24/02/2018

Chapter 1

Introduction

1.1 Problem Statement:

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.

The datasets contains transactions made by credit cards in September 2013 by European cardholders. 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. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Objective:

To identify fraudulent transaction

Overview of the dataset:

Given:

Dataset dimension: 284807 x 31

Attribute Information:

The predictors provided and there datatypes are as follows:

Time – float64

V1 to V28 – float64

Amount – float64

Class – int64 – Target class

Target class:

Class variable is our target class where 1 represent fraudulent transaction and 0 represents non fraudulent transaction.

Give below are the first few observations of the data set :

Table 1.1 Credit Card data set

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...

V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

Chapter 2

Methodology

2.1 Missing value analysis:

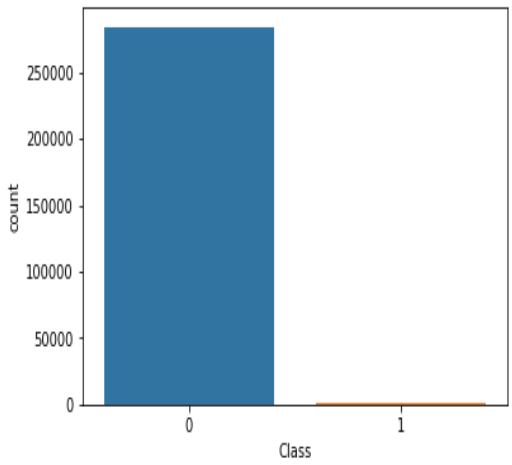
Many times, our data set can have missing values due to various reasons like error in data extraction or data collection .These missing values can be treated in several ways like deleting them or imputing with mean, median, mode ,KNN imputation or make use of other predictive models. However, after analysing we saw that, our data set is free from missing values .

Table 2.1 Missing value analysis for each feature

V1 0.0
V2 0.0
V3 0.0
V4 0.0
V5 0.0
V6 0.0
V7 0.0
V8 0.0
V9 0.0
V10 0.0
V11 0.0
V12 0.0
V13 0.0
V14 0.0
V15 0.0
V16 0.0
V17 0.0
V18 0.0
V19 0.0
V20 0.0
V21 0.0
V22 0.0
V23 0.0
V24 0.0
V25 0.0
V26 0.0
V27 0.0
V28 0.0
Amount 0.0
Class 0.0

2.2 Data visualisation:

Let us see the distribution of Class in the data set. First of all let us know how representative is the data of fraudulent transaction as this data will be provided to the model.



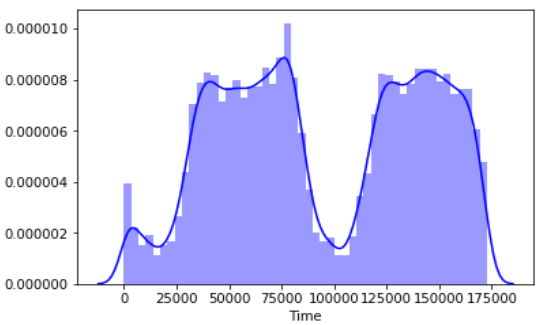
Class	0 (False)	1(True)
	284315	492

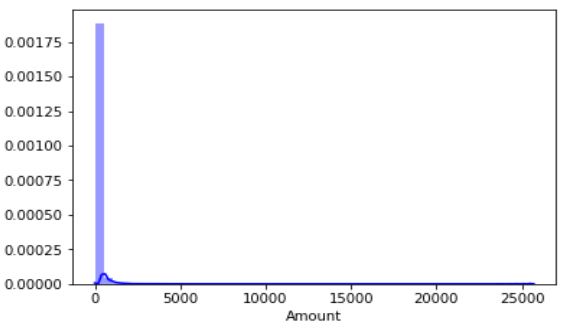
Fraud percent: 492/ (284315) = 0.17 %

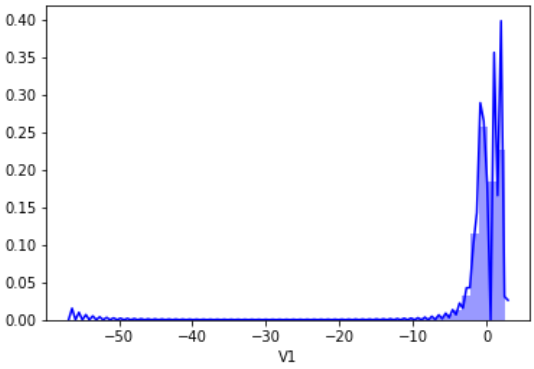
The fraud percent present in our data set is very low, thus our model will not have sufficient use case to predict fraud from this data set. *We will be required to the data set.*

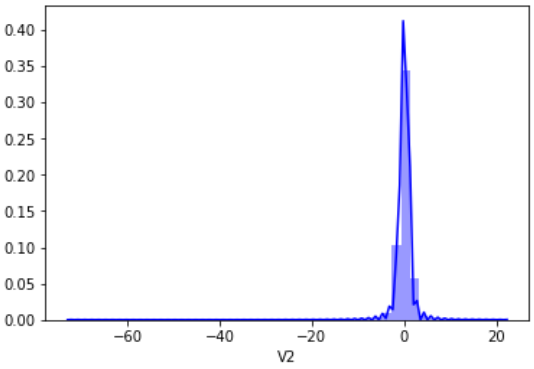
Univariate plots:

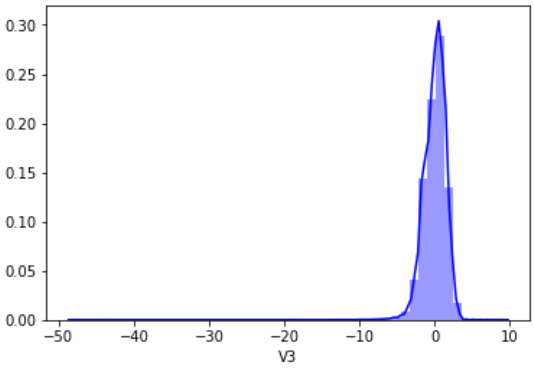
Let us quickly visualise our data set to get the feel of the data distribution and see if we can gain any insights.

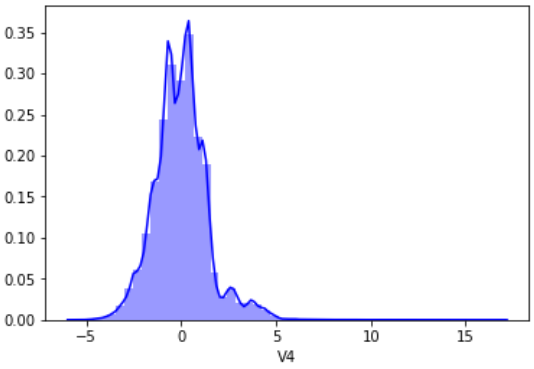


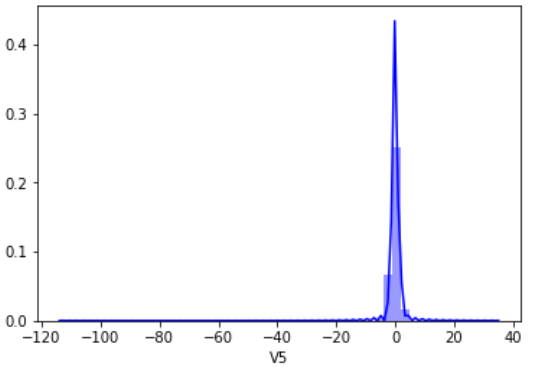


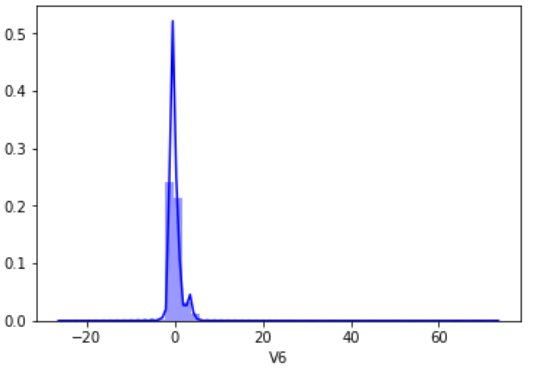

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−50
−40
−30
−20
−10
0
V1

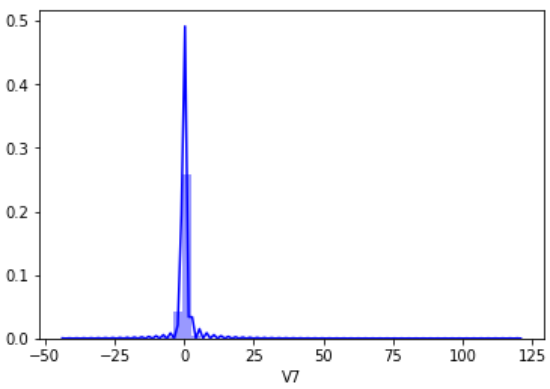

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−60
−40
−20
0
20
V2

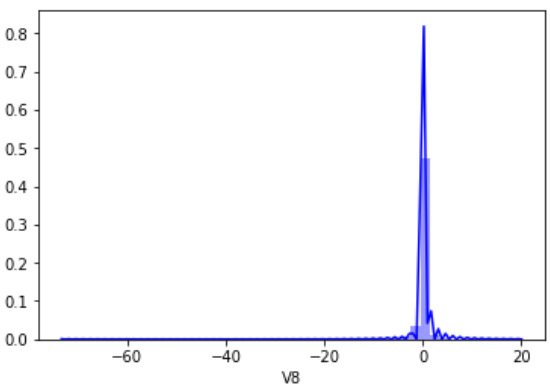

0.30
0.25
0.20
0.15
0.10
0.05
0.00
−50
−40
−30
−20
−10
0
10
V3

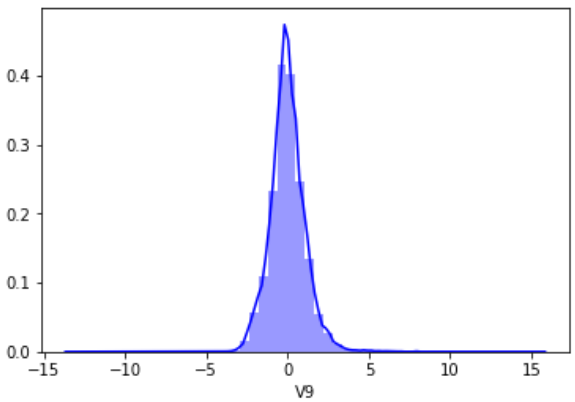

0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15
V4

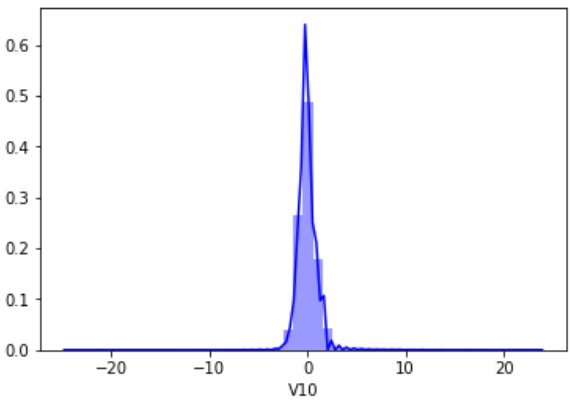

0.4
0.3
0.2
0.1
0.0
−120
−100
−80
−60
−40
−20
0
20
40
V5

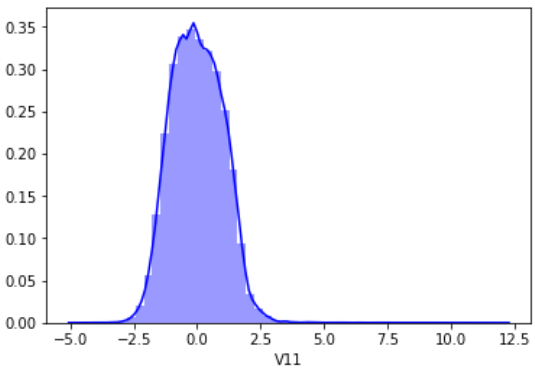

0.5
0.4
0.3
0.2
0.1
0.0
−20
0
20
40
60
V6

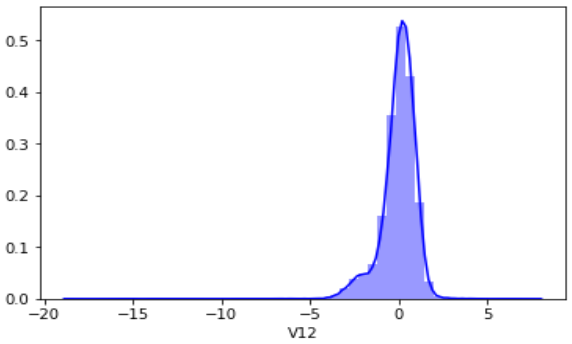

0.5
0.4
0.3
0.2
0.1
0.0
−50
−25
0
25
50
75
100
125
V7

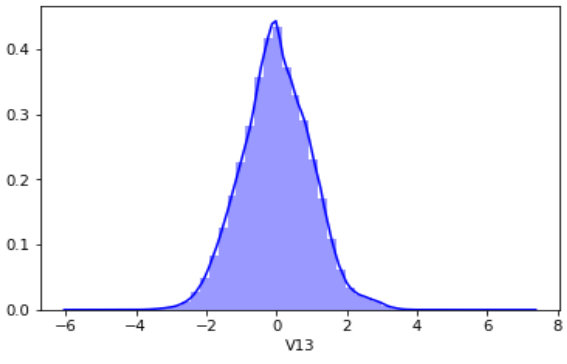

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−60
−40
−20
0
20
V8

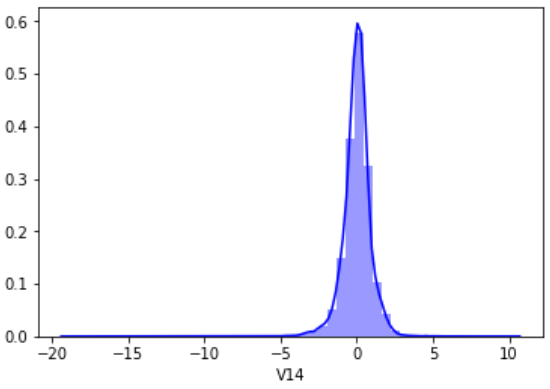

V9

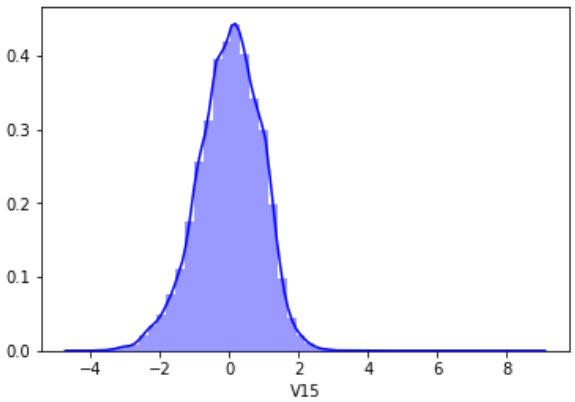

V10

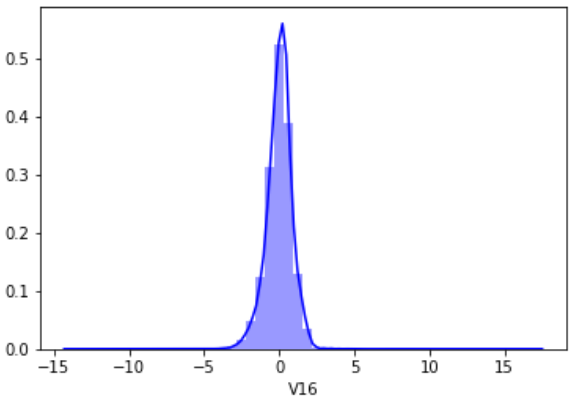

V11

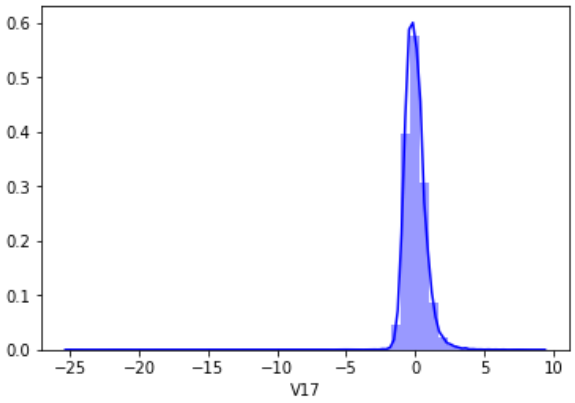

V12

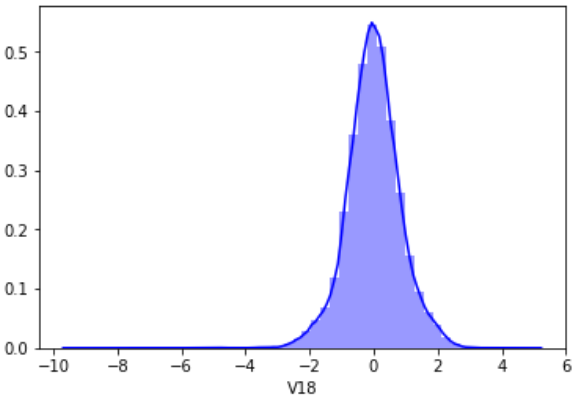

V13

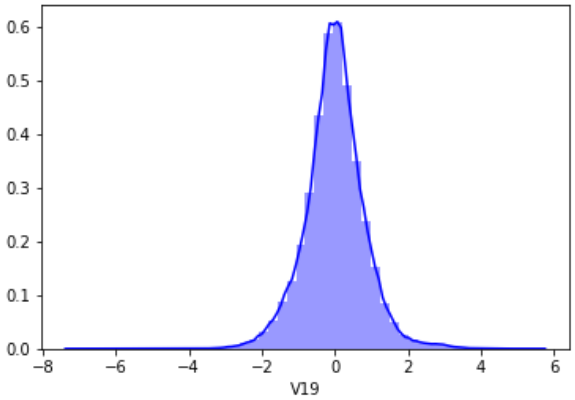

V14

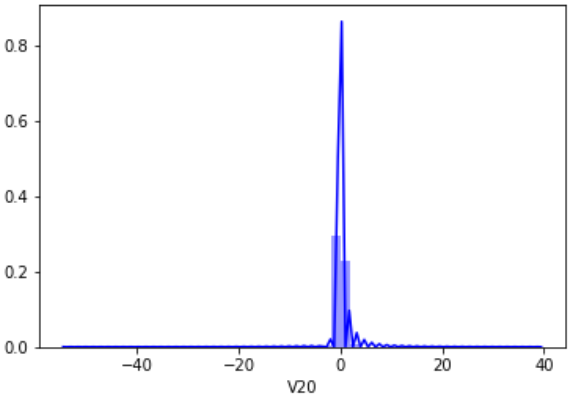

V15

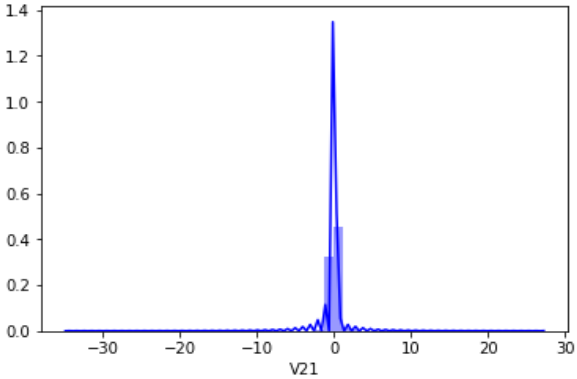

V16

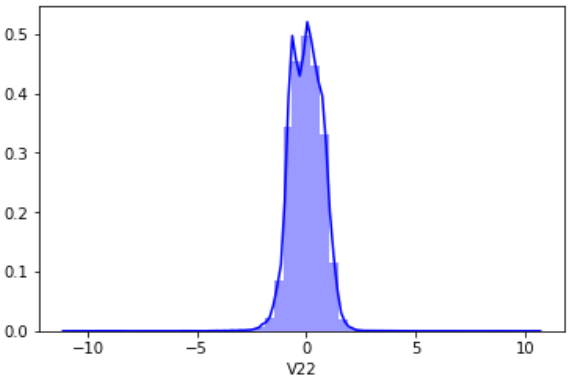

0.6
0.5
0.4
0.3
0.2
0.1
0.0
−25 −20 −15 −10 −5 0 5 10
V17

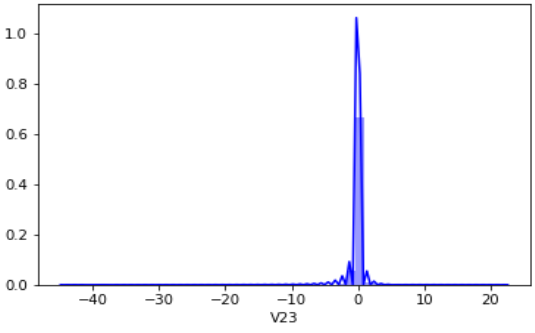

0.5
0.4
0.3
0.2
0.1
0.0
−10 −8 −6 −4 −2 0 2 4 6
V18

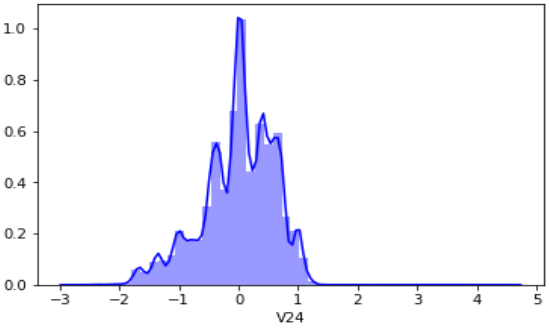

0.6
0.5
0.4
0.3
0.2
0.1
0.0
−8 −6 −4 −2 0 2 4 6
V19

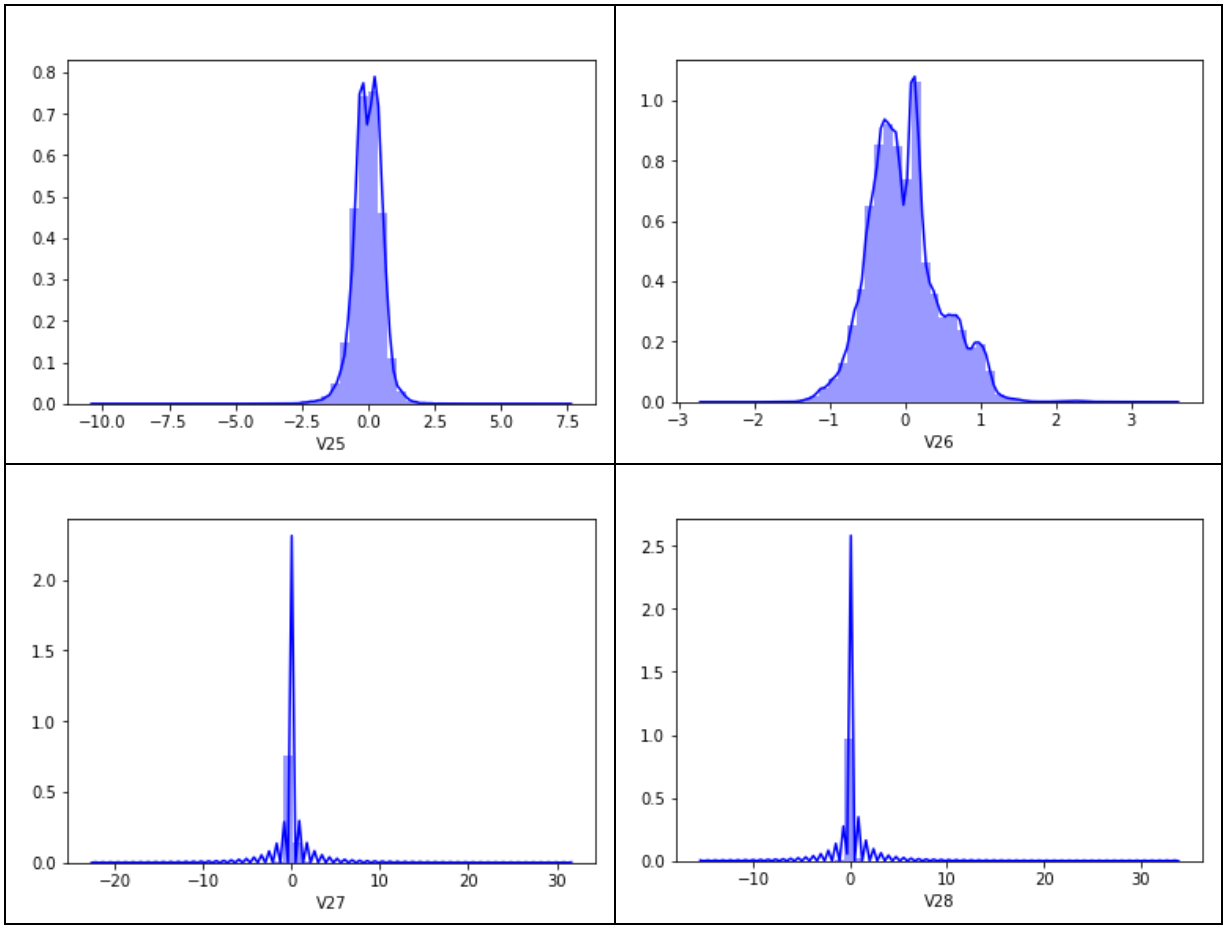

0.8
0.6
0.4
0.2
0.0
−40 −20 0 20 40
V20


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−30 −20 −10 0 10 20 30
V21

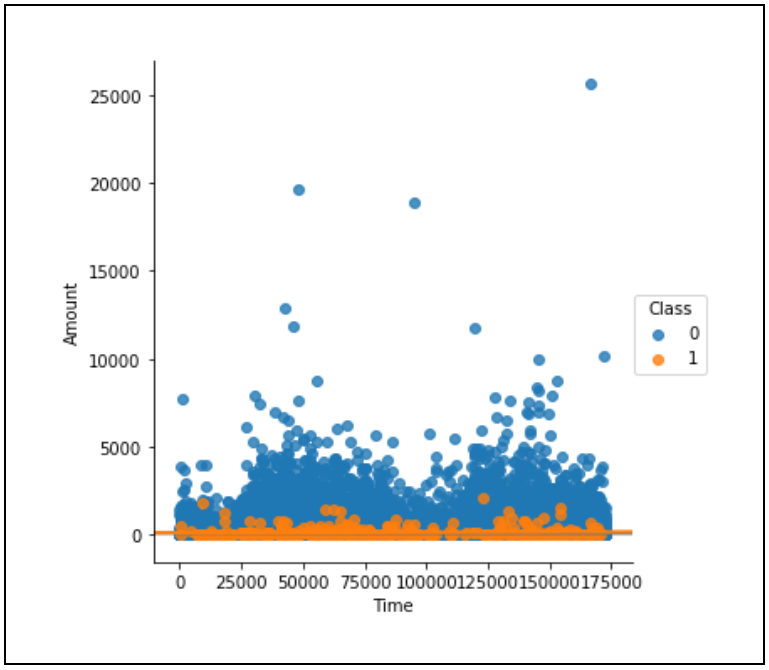

0.5
0.4
0.3
0.2
0.1
0.0
−10 −5 0 5 10
V22


1.0
0.8
0.6
0.4
0.2
0.0
−40 −30 −20 −10 0 10 20
V23


1.0
0.8
0.6
0.4
0.2
0.0
−3 −2 −1 0 1 2 3 4 5
V24



Let us now visualise the distribution of fraudulent transaction with respect to amount and time variable. We see that fraudulent transaction is spread across different times but the key finding here is that the fraudulent transactions are of smaller amount.



2.3 Outlier detection:

Before feeding the data to our model, we would like to analyse outlier in our data set. If not treated they can substantially effect the results of our model. Here, we do outlier analysis for numerical feature set with the help of boxplot method. Any data point that is less than 1.5*IQR (Inter Quartile range) times the 25th percentile or more than 1.5*IQR the 75th percentile, is to be treated as an outlier .We have already seen that our train data set contains only 0.17% of the event rate that we want to predict, thus we should be careful enough to treat these outliers.

In our problem statement, the outliers are capped to 25th and 75th percentile.

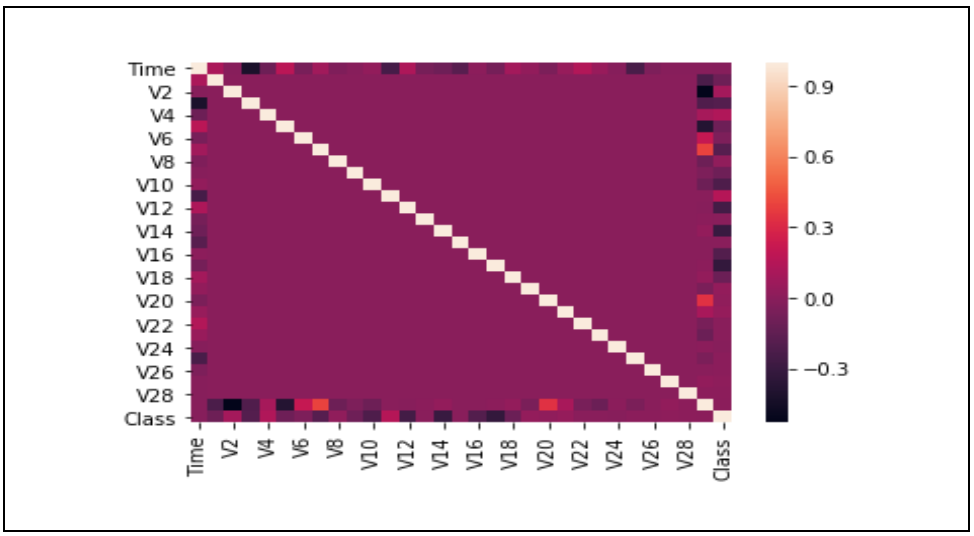
2.4 Feature Selection:

Numerical feature:

For numerical feature set, we perform correlation analysis for feature selection. Correlation tells us how two variables are linearly related to each other. If correlation between two or more variables are high i.e around more than $\pm$0.80, it means that they are carrying same level of information, which suggests that one of the features can be safely dropped. However, variance inflation factor(vif) is a better approach. We will drop features whose vif is more than 10.

Correlation plot:

Table 2.8 Correlation plot of numerical features



Multicollinearity check (Variance inflation factor):

Time	2.896222	V15	1.046080
V1	1.686619	V16	2.388256
V2	2.279518	V17	3.801044
V3	2.147787	V18	1.599216
V4	1.549607	V19	1.132610
V5	1.924199	V20	1.453915
V6	1.493972	V21	1.581428
V7	2.808459	V22	1.457696
V8	1.550857	V23	1.376520
V9	1.506112	V24	1.034379
V10	2.428259	V25	1.237693
V11	1.655301	V26	1.022409
V12	2.823340	V27	1.394548
V13	1.043843	V28	1.433110
V14	2.311255	Amount	3.564293
		Class	2.093847

Feature Scaling:

Before we move further with data modelling, let us scale our data set .From data visualisation plot, we have already seen that most of the columns are centred on zero except Time and Amount.

Standardization of a dataset is a common requirement for many machine learning estimators. Typically this is done by removing the mean and scaling to unit variance. However, outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results.

We have used RobustScaler The centring and scaling statistics of this scaler are based on percentiles and are thus robust to outliers. This Scaler removes the median and scales the data according to the inter-quantile range

$xi = (xi - Q1(x))/(Q3(x) - Q1(x))$

2.5 Model Development

Let us now try different machine learning algorithms for classification. The model is built through cross validation approach.

We will be using SMOTE to balance our imbalanced data set.

The key point here is to balance data set along with cross validation and not before otherwise we will be fooling our cross validation set with the wrong data set.

We start with the most basic algorithm for classification then move towards the complex ones.

We have created a function "cross_val_performance" which evaluates our model. The different error metric used are:

- Accuracy
- Precision
- Recall
- F1 score
- AUC -ROC

Out of all these metrics ,we will be mainly selecting our model on the basis of Recall error metric as all the actual fraudulent transaction must be correctly classified to prevent customer from losing money.

Let us briefly have an overview of what the error metrics mean.

Confusion matrix:

Predicted Class / Actual Class	Negatives(0)	Positives(1)
Negatives(0)	True Negative(TN)	False Positive(FP)
Positives(1)	False Negative(FN)	True Positive(TP)

Accuracy: It is the proportion of total number of predictions correctly classified. It is a good measure when the classes are equally distributed.
= (TP+TN)/ (TP+TN+FP+FN)

Precision : The proportion of predicted positive cases that were correctly

classified.
= TP/(TP+FP)

Recall or Sensitivity: Actual Positive classes that were correctly classified by the model.
= TP/TP+FN

F1 Score: We don't really want to carry both Precision and Recall in our pockets every time we make a model for solving a classification problem. So it's best if we can get a single score that kind of represents both Precision(P) and Recall(R).
One way to do that is simply taking their arithmetic mean. i.e (P + R) / 2 where P is Precision and R is Recall. But that's pretty bad in some situations. As the score will be effected by large values. However, if we take harmonic mean, the score is effected by small values and less sensitive to large values. Thus penalising poor models heavily.
So, we need something more balanced than the arithmetic mean and that is harmonic mean.

= 2 * Precision * Recall / (Precision + Recall)

AUC-ROC: An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$$TPR=TP/(TP+FN)$$

False Positive Rate (FPR) is defined as follows:

$$FPR=FP/(FP+TN)$$

An ROC curve plots TPR vs. FPR at different classification thresholds.

SMOTE(Synthetic Minority Oversampling Technique) :
To balance our imbalance data set we apply SMOTE technique
Let us now go through different machine learning algorithms. In the output section, we will evaluate the mean performance of the k fold cross validation approach.

Logistic Regression:

Code:

```
model_LR = LogisticRegression(penalty="l1",solver = 'liblinear')
cross_val_performance(train_set,model_LR)
model_performance(test_y,model_LR.predict(test_x))
```

Output:

```
Mean score of k fold  Cross Validation :

Accuracy mean: 0.969270382248
Precision mean: 0.0490555070842
Recall mean: 0.881578947368
F1 mean: 0.0929076210392

Score on test data :

accuracy score: 0.967191932811
precision: 0.0431211498973
recall: 0.945945945946
f1: 0.082482325216
```

Decision Trees:

Code:

```
model_DT =
DecisionTreeClassifier(max_depth=10,criterion="gini",random_state=1)
cross_val_performance(train_set,model_DT)
model_performance(test_y,model_DT.predict(test_x))
```

Output:

```
Mean score of k fold  Cross Validation :

Accuracy mean: 0.977480537699
Precision mean: 0.077621607996
Recall mean: 0.862310875469
F1 mean: 0.139847682694
```

```
Score on test data :

accuracy score: 0.984354372068
precision: 0.0789252728799
recall: 0.846846846847
f1: 0.144393241167
```

Random Forest:

Code:

```
param_RF = {"max_depth": [3, None],"max_features": sp_randint(1,
11),"min_samples_split": sp_randint(2, 11),"bootstrap": [True,
False],"criterion": ["gini", "entropy"]}
model_RF =
RandomizedSearchCV(RandomForestClassifier(),param_RF,cv=5)
cross_val_performance(train_set,model_RF)
model_performance(test_y,model_RF.predict(test_x))
```

Output:

```
Mean score of k fold  Cross Validation :

Accuracy mean: 0.985928046984
Precision mean: 0.381393649954
Recall mean: 0.82161034243
F1 mean: 0.390830881165

Score on test data :
accuracy score: 0.999564618971
precision: 0.908163265306
recall: 0.801801801802
f1: 0.851674641148
```

SVC (Support Vector Classifier):

Code:

```
model_SVC = SVC(kernel="linear",C=1,gamma=1)
cross_val_performance(train_set,model_SVC)
model_performance(test_y,model_SVC.predict(test_x))
```

Output:

```
Mean score of k fold  Cross Validation :

Accuracy mean: 0.983107333532
Precision mean: 0.335908400493
Recall mean: 0.832285406003
F1 mean: 0.349519313303

Score on test data :

accuracy score: 0.967093620966
precision: 0.0426229508197
recall: 0.936936936937
f1: 0.0815366522932
```

XGBoost :

Code:

```
model_XGBoost = XGBClassifier(silent=False,
scale_pos_weight=1,learning_rate=0.01,colsample_bytree =
0.4,subsample = 0.8,objective='binary:logistic',
n_estimators=10,reg_alpha = 0.3,max_depth=4, gamma=10)

model_XGBoost.fit(train_x,train_y)

model_performance(test_y,model_XGBoost.predict(test_x))
```

Output:

```
Score on train data :

accuracy score: 0.999227546172
precision: 0.852941176471
recall: 0.685039370079
f1: 0.759825327511

Score on test data :

accuracy score: 0.999410128929
precision: 0.887640449438
recall: 0.711711711712
f1: 0.79
```

3. Conclusion

Model Evaluation and Model Selection:

In the given context of our problem statement we wanted to reduce fraudulent transaction. Hence, it is very important to classify actual fraudulent transaction correctly. Thus in the give scenario recall is a better error metric for model selection and we should not be misled by accuracy score of the model.

Machine Learning Model	Precision	Recall
Logistic Regression	4.31%	**94.59%**
Decision Tree	7.89%	84.68%
Random Forest	**90.81%**	80.18%
Support Vector Classifier	4.26%	93.69%
XGBoost	88.76%	71.17%

We can see that Random Forest with hyperparameters tuned using Random Search Cross Validation performs best in terms of Precision and Logistic Regression with 'L1' Regularisation in terms of Recall .

References:

https://www.analyticsvidhya.com

https://www.kdnuggets.com/

https://towardsdatascience.com/

Note: Figures and References are made from Python code outputs