CREDIT CARD FRAUD DETECTION

-Project work by Sreya Bhattacharya

SUBMITTED FOR DATA SCIENCE TASK IN CODSOFT FOR INTERNSHIP

Introduction:

A credit card is a type of credit facility, provided by banks that allow customers to borrow funds within a pre-approved credit limit. It enables customers to make purchase transactions on goods and services.
So 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. This project is an application of machine learning with R programming.

Data Source:

I've collected the dataset from the given link in the task.
(https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

Data Description:

The dataset contains transactions made by credit cards in September 2013 by European cardholders.
From the information of Kaggle website we know "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, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise."

Objective:

Here in this project I want to build a machine learning model to identify fraudulent credit card transactions based on the given features (V1, V2,…..,V28). Here I'll divide the dataset into Train & Test data (80:20 ratio). I'll fit a logistic regression model in the Train Dataset and I'll check the accuracy of my prediction in the Test Dataset.

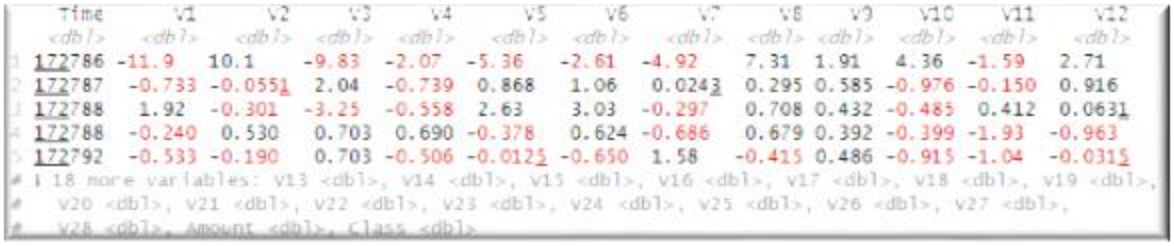
Calculation & Analysis:

We have used **R Studio** for all the calculations and analysis.

- First, we load the data at R Studio. i.e.-

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	0	-1.36	-0.0728	2.54	1.38	-0.338	0.462	0.240	0.0987	0.364	0.0908	-0.552
2	0	1.19	0.266	0.166	0.448	0.0600	-0.0824	-0.0788	0.0851	-0.255	-0.167	1.61
3	1	-1.36	-1.34	1.77	0.380	-0.503	1.80	0.791	0.248	-1.51	0.208	0.625
4	1	-0.966	-0.185	1.79	-0.863	-0.0103	1.25	0.238	0.377	-1.39	-0.0550	-0.226
5	2	-1.16	0.878	1.55	0.403	-0.407	0.0959	0.593	-0.271	0.818	0.753	-0.823
6	2	-0.426	0.961	1.14	-0.168	0.421	-0.0297	0.476	0.260	-0.569	-0.371	1.34
7	4	1.23	0.141	0.0454	1.20	0.192	0.273	-0.00516	0.0812	0.465	-0.0993	-1.42
8	7	-0.644	1.42	1.07	-0.492	0.949	0.428	1.12	-3.81	0.615	1.25	-0.619
9	7	-0.894	0.286	-0.113	-0.272	2.67	3.72	0.370	0.851	-0.392	-0.410	-0.705
10	9	-0.338	1.12	1.04	-0.222	0.499	-0.247	0.652	0.0695	-0.737	-0.367	1.02

It's the first 10 rows of the data &



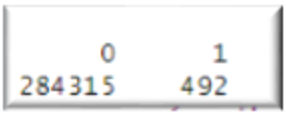
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	172786	-11.9	10.1	-9.83	-2.07	-5.36	-2.61	-4.92	7.31	1.91	4.36	-1.59	2.71
2	172787	-0.733	-0.0551	2.04	-0.739	0.868	1.06	0.0243	0.295	0.585	-0.976	-0.150	0.916
3	172788	1.92	-0.301	-3.25	-0.558	2.63	3.03	-0.297	0.708	0.432	-0.485	0.412	0.0631
4	172788	-0.240	0.530	0.703	0.690	-0.378	0.624	-0.686	0.679	0.392	-0.399	-1.93	-0.963
5	172792	-0.533	-0.190	0.703	-0.506	-0.0125	-0.650	1.58	-0.415	0.486	-0.915	-1.04	-0.0315

i 18 more variables: V13 <dbl>, V14 <dbl>, V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>,
V20 <dbl>, V21 <dbl>, V22 <dbl>, V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>,
V28 <dbl>, Amount <dbl>, Class <dbl>

It's the last 5 rows of the given data.

Then the calculations and the analysis are done as follows,

- We'll check how many credit cards are fraud as per the given data.



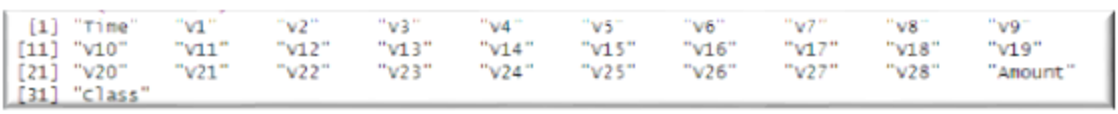
0	1
284315	492

So, 492 credit cards are fraud as per data information.

- Next, we'll summarize the dataset. We get,

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	5.60	22.00	88.35	77.17	25691.16

- Now we'll check the variable's names in the given data, that is-



```
 [1] "Time"   "V1"   "V2"   "V3"   "V4"   "V5"   "V6"   "V7"   "V8"   "V9"
[11] "V10"    "V11"  "V12"  "V13"  "V14"  "V15"  "V16"  "V17"  "V18"  "V19"
[21] "V20"    "V21"  "V22"  "V23"  "V24"  "V25"  "V26"  "V27"  "V28"  "Amount"
[31] "Class"
```

- We'll calculate the variance and SD of the Amount from the dataset.
  - ❍ Variance : 62560.07
  - ❍ Standard Deviation : 250.1201

- Next we'll eliminate the 1st column (i.e. - time) as it's not needed. So the new dataset is-

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
-1.36	-0.0728	2.54	1.38	-0.338	0.462	0.240	0.0987	0.364	0.0908	-0.552	-0.618	-0.991
1.19	0.266	0.166	0.448	0.0600	-0.0824	-0.0788	0.0851	-0.255	-0.167	1.61	1.07	0.489
-1.36	-1.34	1.77	0.380	-0.503	1.80	0.791	0.248	-1.51	0.208	0.625	0.0661	0.717
-0.966	-0.185	1.79	-0.863	-0.0103	1.25	0.238	0.377	-1.39	-0.0550	-0.226	0.178	0.508
-1.16	0.878	1.55	0.403	-0.407	0.0959	0.593	-0.271	0.818	0.753	-0.823	0.538	1.35
-0.426	0.961	1.14	-0.168	0.421	-0.0297	0.476	0.260	-0.569	-0.371	1.34	0.360	-0.358

i 17 more variables: V14 <dbl>, V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>, V22 <dbl>, V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl[,1]>, Class <dbl>

- Next I'll split the dataset into train & test dataset. Let us check the dimensions of the datasets.

```
> dim(train_data)
[1] 227846     30
> dim(test_data)
[1] 56961    30
```

- **Fitting the Logistic Model:**

Now we'll fit a logistic regression model into the Train dataset. The estimate of the coefficients and the p-values are given below.

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6108  -0.0292  -0.0194  -0.0125   4.6021

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.651305   0.160212 -53.999  < 2e-16 ***
V1           0.072540   0.044144   1.643 0.100332
V2           0.014818   0.059777   0.248 0.804220
V3           0.026109   0.049776   0.525 0.599906
V4           0.681286   0.078071   8.726  < 2e-16 ***
V5           0.087938   0.071553   1.229 0.219079
V6          -0.148083   0.085192  -1.738 0.082170 .
V7          -0.117344   0.068940  -1.702 0.088731 .
V8          -0.146045   0.035667  -4.095 4.23e-05 ***
V9          -0.339828   0.117595  -2.890 0.003855 **
V10         -0.785462   0.098486  -7.975 1.52e-15 ***
V11          0.001492   0.085147   0.018 0.986018
V12          0.087106   0.094869   0.918 0.358532
```

```
V13          -0.343792   0.092381  -3.721 0.000198 ***
V14          -0.526828   0.067084  -7.853 4.05e-15 ***
V15          -0.095471   0.094037  -1.015 0.309991
V16          -0.130225   0.138629  -0.939 0.347537
V17           0.032463   0.074471   0.436 0.662900
V18          -0.100964   0.140985  -0.716 0.473909
V19           0.083711   0.105134   0.796 0.425897
V20          -0.463946   0.081871  -5.667 1.46e-08 ***
V21           0.381206   0.065880   5.786 7.19e-09 ***
V22           0.610874   0.142086   4.299 1.71e-05 ***
V23          -0.071406   0.058799  -1.214 0.224589
V24           0.255791   0.170568   1.500 0.133706
V25          -0.073955   0.142634  -0.519 0.604109
V26           0.120841   0.202553   0.597 0.550783
V27          -0.852018   0.118391  -7.197 6.17e-13 ***
V28          -0.323854   0.090075  -3.595 0.000324 ***
Amount        0.292477   0.092075   3.177 0.001491 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5799.1  on 227845  degrees of freedom
Residual deviance: 1790.9  on 227816  degrees of freedom
AIC: 1850.9

Number of Fisher Scoring iterations: 12
```

The coefficients in the output indicate the average change in log odds of class. For example, a one unit increase in **V4** is associated with an average increase of **2e-16** in the log odds of Class.

The p-values in the output gives an idea of how effective each predictor variable is at predicting the probability of Class. Here we can see that V4, V8, V9, V10, V13, V14, V20, V21, V22, V27, V28 are important predictors since they have low p-values (<0.05) while the others are not nearly as important.

Now with these important covariates we again fit the logistic regression model. And the summary is as follows,

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.4178  -0.0293  -0.0196  -0.0129   4.8567

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.59047    0.14038 -61.196  < 2e-16 ***
V4           0.64076    0.04983  12.859  < 2e-16 ***
V8          -0.14204    0.02017  -7.041 1.91e-12 ***
V9          -0.22355    0.08464  -2.641 0.008260 **
V10         -0.73948    0.06043 -12.236  < 2e-16 ***
V13         -0.33284    0.08578  -3.880 0.000104 ***
V14         -0.54448    0.04103 -13.269  < 2e-16 ***
V20         -0.28292    0.04946  -5.720 1.07e-08 ***
V21          0.42856    0.04894   8.758  < 2e-16 ***
V22          0.71467    0.11748   6.083 1.18e-09 ***
V27         -0.65176    0.10074  -6.470 9.83e-11 ***
V28         -0.28006    0.08381  -3.342 0.000833 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see that here all the predictors are important predictors as the p-values are very low (<0.05).

- **Assessing Model Fit**

For logistic regression we compute a metric known as **McFadden's R^2**, which ranges from 0 to just under 1. Values close to 0 indicate that the model has no predictive power. In practice, values over 0.40 indicate that a model fits the data very well.

We can compute McFadden's R^2 for our model using the **pR²** function from the pscl package. The McFadden's R^2 for the train data is:

```
fitting null model for pseudo-r2
 McFadden
0.6842621
```

This is a high value for McFadden's R^2, which indicates that our model fits the data very well and has high predictive power.

- **Variable Importance**

Higher values indicate more importance. These results match up nicely with the p-values from the model.

	Overall
V4	12.858599
V8	7.041056
V9	2.641263
V10	12.236165
V13	3.880130
V14	13.269029
V20	5.719644
V21	8.757666
V22	6.083315
V27	6.469590
V28	3.341738

So, from the values we can clearly see that v14 is most important predictor followed by others.

- **Multicollinearity Checking**

VIF Values

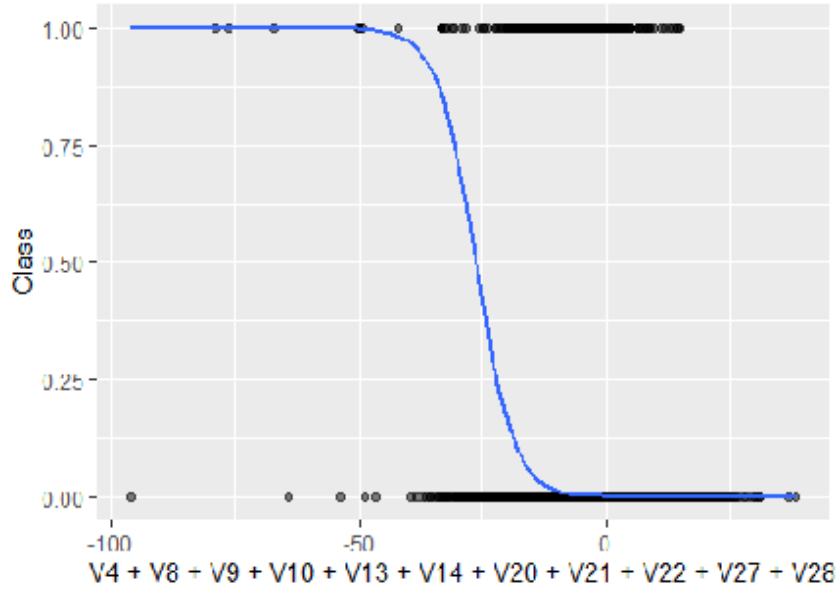
We calculate the VIF values of each variable for train data in the model to check the Multicollinearity.

V4	V8	V9	V10	V13	V14	V20	V21	V22	V27	V28
1.680487	1.158976	2.465486	2.942776	1.136610	2.081252	2.137689	1.967048	2.076763	2.322942	1.169175

As a rule of thumb, VIF values above 5 indicate severe multicollinearity. Since none of the predictor variables in our models have a VIF over 5, we can assume that multicollinearity is not an issue in our model.

- **Plotting the Data**

We plot our train data by using ggplot, which is used to construct the initial plot object, and is almost always followed by a plus sign (+) to add components to the plot. A logistic model shows **S shaped** curve, called **Sigmoid**. The model fitted on the train data also shows an S shaped curve or sigmoid curve. From here we can tell that, multiple logistic regression model fits our data very well. The plotted curve is given below,



- **Using the Model to Make Predictions**

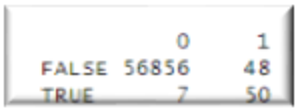
We have fitted the logistic regression model to the train data. Now we will predict the response variable (i.e. -Class) for the test data based on the model fitted to the train data. The prediction of the Class is made based on the covariates V4, V8, V9, V10, V13, V14, V20, V21, V22, V27 and V28.

- **Model Diagnosis**

Now, we analyse how well our model performs on the test data.

- **Confusion Matrix**

We create confusion matrix which shows our predictions compared to the actual Class. The confusion matrix is as follows,



	0	1
FALSE	56856	48
TRUE	7	50

From the confusion matrix we'll calculate sensitivity, specificity, precision, F1 value and accuracy of the data.

- **Sensitivity**

We calculate sensitivity (which is also known as recall) from confusion matrix and the value for our trend data is 0.877193. It is high so our model is able to predict the outcomes. So, this particular model turns out to be very good at predicting whether a class is malignant or benign.

- **Specificity**

The specificity for our test data is **0.999157** which indicates that this particular model fits our data very well.

- **Precision**

The precision tells us about the quality of positive predictions. The value of precision for this dataset is **0.999876.**

- **F1 Value**

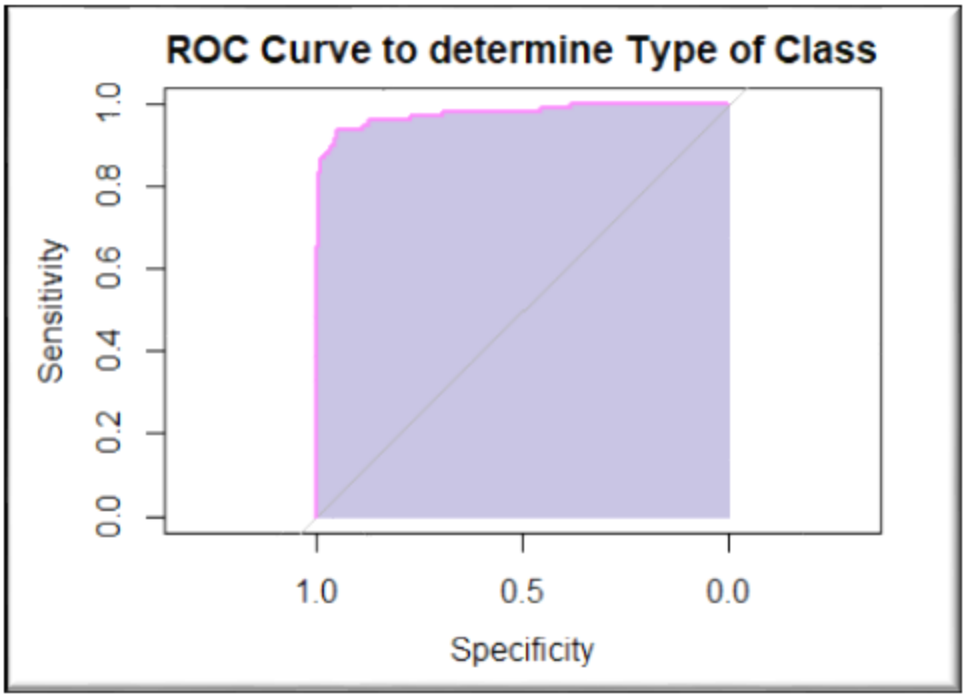
F1 score is a measure of the harmonic mean of precision and sensitivity (recall). Here the measurement of F1-value is **0.934526,** which is very high. Therefore we can see that our model performs well.

- **Accuracy**

The accuracy for our data is **0.999034**, which is very high. In another way beside total misclassification error rate by accuracy we can say that, the logistic regression model fits the data very well.

- **ROC Curve**

Lastly, we can plot the ROC (Receiver Operating Characteristic) Curve which displays the percentage of true positives predicted by the model as the prediction probability cutoff is lowered from 1 to 0. AUC (the area under the curve) for train data is **0.9774**, which is very high. This indicates that our model does a good job of predicting the Class of an individual. The ROC curve is as follows,


ROC Curve to determine Type of Class
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity

Conclusion:

I've divided the dataset in two parts "Train" & "Test". The Train dataset contains 227846 observations and the Test dataset contains 56961 observations.

Here we fitted a Logistic Regression model in our train dataset and based on the fitted model we made predictions in our Test dataset and from the calculations we can see that our prediction is 99.90345% accurate**.**

Reference:

1. Fundamentals of Machine Learning for Predictive Data Analytics by John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy.

2. Programming Collective Intelligence by Toby Segaran

3. Hands-On Programming with R by Garrett Grolemund & Hadley Wickham

4. R in action by Dr. Robert L. Kabacoff.

Acknowledgement:

I would like to express my special thanks of gratitude to CODSOFT for giving me the golden opportunity to do the project on the wonderful topic which helped me doing a lot of research and I came to know about so many things. It helped me increase my knowledge and skills.

Annexure:

```
library(caret)
library(data.table)
dim(creditcard)
head(creditcard,10)
tail(creditcard,5)
table(creditcard$Class)
summary(creditcard$Amount)
names(creditcard)
var(creditcard$Amount)
sd(creditcard$Amount)
creditcard$Amount=scale(creditcard$Amount)
NewData=creditcard[,-c(1)]
head(NewData)
library(caTools)
set.seed(123)
data_sample = sample.split(NewData$Class,SplitRatio=0.80)
train_data = subset(NewData,data_sample==TRUE)
test_data = subset(NewData,data_sample==FALSE)
dim(train_data)
dim(test_data)
Logistic_Model=glm(Class~.,train_data,family=binomial())
summary(Logistic_Model)
Model<-
glm(Class~V4+V8+V9+V10+V13+V14+V20+V21+V22+V27+V28+Amount[,
1],family=binomial,data=  train_data)
summary(Model)
```

```
model<-
glm(Class~V4+V8+V9+V10+V13+V14+V20+V21+V22+V27+V28,family=bi
nomial,data= train_data)

summary(model)

library(pscl)

pscl::pR2(model)["McFadden"]

library(caret)

caret::varImp(model)

library(car)

car::vif(model)

library(caret)

library(ggplot2)

library(lattice)

ggplot(model,aes(x=V4+V8+V9+V10+V13+V14+V20+V21+V22+V27+V28,
y=Class)) + geom_point(alpha=.5) +stat_smooth(method="glm", se=FALSE,
method.args = list(family=binomial))

predicted<-predict(model,test_data,type="response")

summary(predicted)

confusion_matrix<-table(predicted>0.5,test_data$Class)

confusion_matrix

sensitivity <- confusion_matrix[2, 2] / (confusion_matrix[2, 2] +
confusion_matrix[2, 1])

sensitivity

Specificity <- confusion_matrix[1, 1] / (confusion_matrix[1, 1] +
confusion_matrix[1, 2])

specificity

precision<-confusion_matrix[1, 1] / (confusion_matrix[1, 1] +
confusion_matrix[2, 1])

precision

F1score<-2*precision*sensitivity/(precision+sensitivity)
```

F1score

```
accuracy<-sum(diag(confusion_matrix))/sum(confusion_matrix)

print(accuracy

library(pROC)

roc_obj<-roc(test_data$Class,predicted)

auc<-auc(roc_obj)

print(auc)

plot(roc_obj,col="violet",type="shape",auc.polygon=TRUE,auc.polygon.col=rgb(0.35,0.31,0.61,alpha=0.4),auc.polygon.border=rgb(0.35,0.31,0.61,0.4),main="ROC Curve to determine Type of Class")
```