CY Tech Sciences et Techniques

Credit Card Fraud Detection

Author
Buu NGUYEN
Anh-Thu DOAN
Cornelis CASPER

December 2021

Contents

1 Introduction **2**

2 Methods **2**

3 Results **4**
3.1 Explanatory Analysis 4
3.2 Classification Model 9

4 Discussion **10**

References **11**

Appendices **11**

A Explanatory Analysis notebook **11**

B Machine Learning notebook **20**

List of Figures

1 Distribution of transactions class 4
2 Distribution of transactions amount 5
3 Transactions amount during the time of day 6
4 Transaction time by Class 7
5 Transaction amount by Class 8
6 Confusion Matrix 9

1 Introduction

A credit card is a thin rectangular piece of plastic or metal issued by a bank or financial services company that allows cardholders to borrow funds with which to pay for goods and services with merchants that accept cards for payment. Credit cards impose the condition that cardholders pay back the borrowed money, plus any applicable interest, as well as any additional agreed-upon charges, either in full by the billing date or over time (Bloomenthal, 2021). Nowadays, the information on the card is read by ATMs, banks and is also used in online internet banking. They contain a unique card number and a secret code. The security relies on the physical safety of the plastic card and the privacy of its number and information. Credit card transactions has increased year by year, leading to an increase in fraudulent activity. As a result, credit card fraud has become a common term. It is a fraud theft committed using a credit card as a fraudulent source of money in a particular transaction.

Credit card fraud occurs when one person uses another person's card without the card owner's knowledge of this activity. When such cases arise, they will be used until the total available limit is exhausted. Numerous fintech companies have developed complicated systems to detect credit card fraud. In addition, most detection systems are based on artificial intelligence and pattern recognition. Furthermore, many intermediary efficient and secure electronic platforms (Payconiq, Paypal, etc.) are on the market to replace traditional credit cards for online payment.

This study focuses on credit card fraud and how to detect them. Our goal is to develop an efficient detection model that can be applied in production. Thus, we need a solution that minimizes the threat of credit card fraud and is also not too aggressive to block legitimate transactions. Therefore, we emphasize developing an efficient and secure system for detecting fraudsters.

2 Methods

The data was collected from over 285,000 anonymized transactions made by credit cards in September 2013 by European cardholders. This dataset was highly unbalanced: the fraud transactions accounted for 0.17% of all transactions. There were 31 features in our dataset:

- Features **V1** to **V28** were the principal components obtained with PCA. Unfortunately, the original features and more background information about the data were not provided due to confidentiality and privacy issues.

- **Time** contained the seconds elapsed between each transaction and the first transaction in the dataset. It was converted to **Hour** to identify if the transaction's hour was correlated with the possibility of a fraudulent transaction.
- **Amount** was the transaction amount.
- **Class** was the response label where 0 denoted a genuine transaction and 1 signified fraud.

R and its libraries (tidyverse, ggplot2, etc) were used to explore the interactions and correlations between features then visualize them. Python were used for the machine learning part in this study.

Since the dataset was highly unbalanced, using **Accuracy** as the metric to evaluate the efficiency of machine learning models was not a good idea. In this study, the fraud transactions were more critical, and Type II errors were more costly. Therefore, **F2 score** was chosen as the primary metric of our models.

$$F_2 = \frac{TP}{TP + 0.2FP + 0.8FN}$$

A pipeline was built to evaluate machine learning models using the scikit-learn library. First, Amount and Hour variables were normalized using Standard Scaler. Then, Logistic Regression, Decision Tree, and Random Forest models were selected for this classification problem. The validation strategy was K-fold cross-validation with five folds and repeated three times. The GridSearchCV method was used to tune models' hyperparameters.

3 Results

3.1 Explanatory Analysis

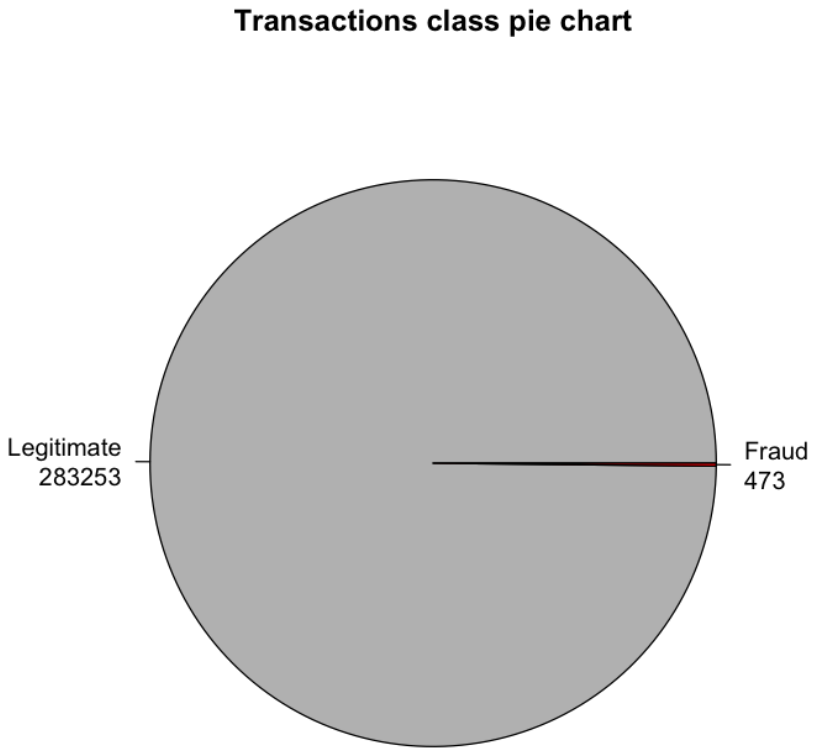


Figure 1: Distribution of transactions class

It is apparently seen from Figure 1 that there were a huge different between two values in Class variable. The value **0** which represented a legit transaction took up almost 99.83% while the value **1** which perform the illegal transaction only held 0.17%.

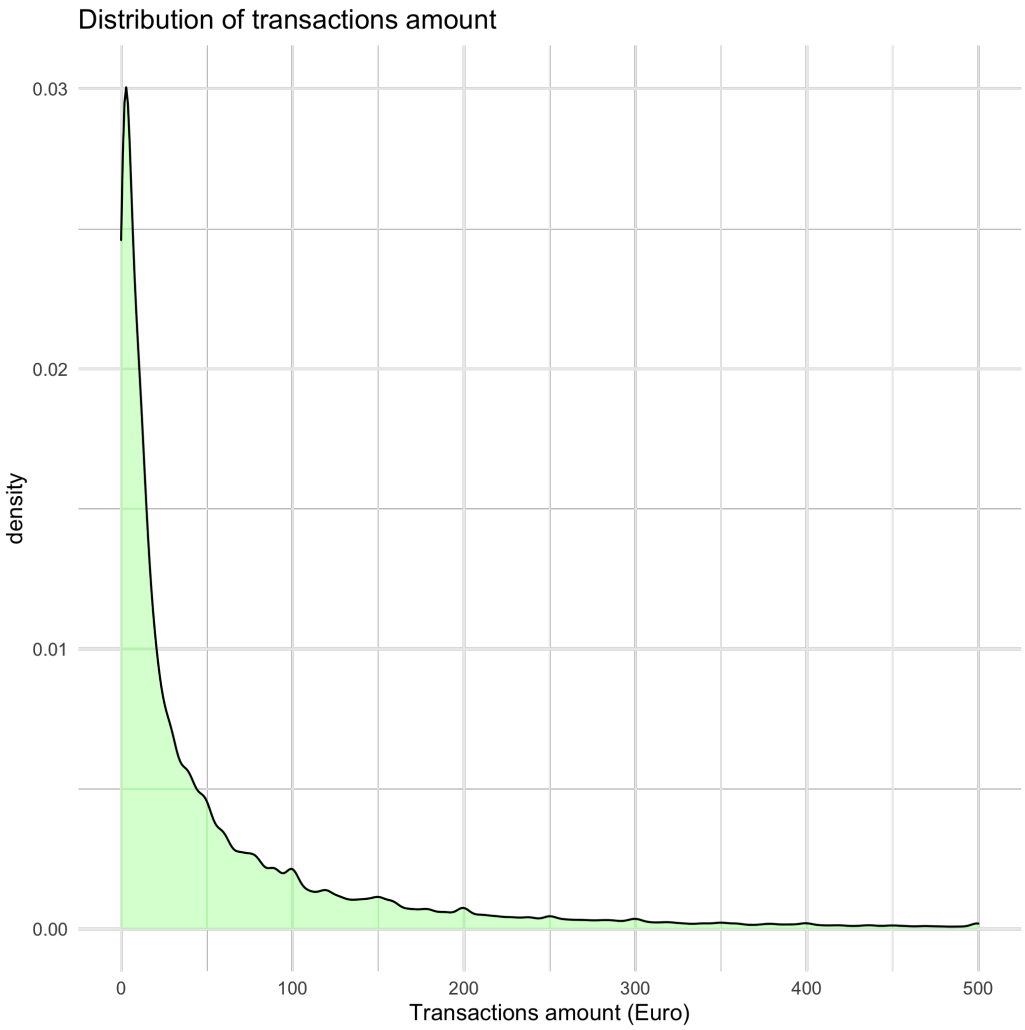


Figure 2: Distribution of transactions amount

	Fraud	mean(Amount)	median(Amount)	sd(Amount)
1	Fraudulent	122.21	9.25	256.68
2	Legitimate	88.29	22.00	250.11

Table 1: Descriptive statistics table of transactions class

As we can see from the Figure 2 and Table 1, the distribution is shifted to the left of the median. We can observe that 80% of the amounts are between 0 and 100 euros. The average of the Legitimate transaction which is 80 euros, higher than the median which is 22 euros for this class. Meanwhile

the median of the Fraudulent transaction is 9.25 euros and the mean is 123 euros (higher than the genuine transaction 35 euros).

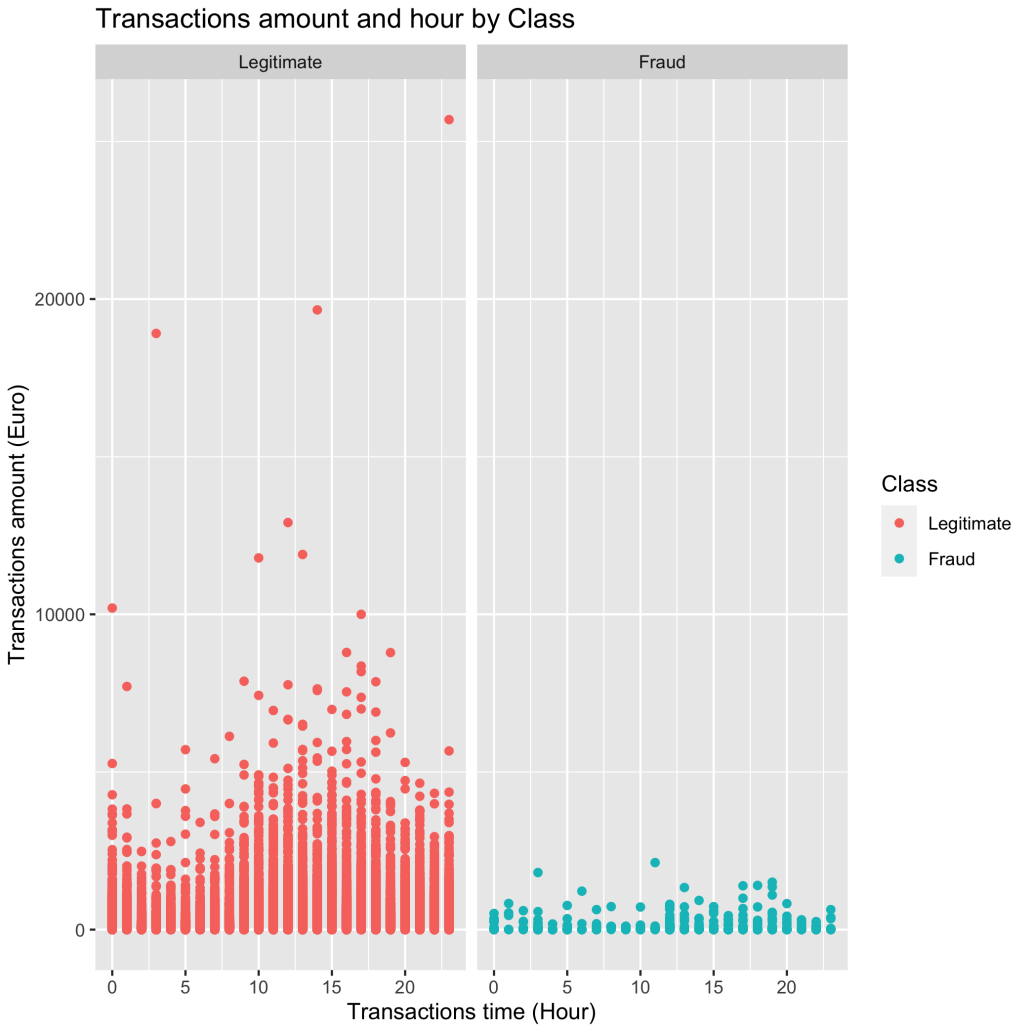


Figure 3: Transactions amount during the time of day

In the given Figure 3, the legitimate transaction decreased throughout the night from 1 am to 6 am, and it increased at the beginning of the day and kept remaining the whole day. On the other hand, the rate of the illegal transactions amount is more likely to occur during the early hours of a day.

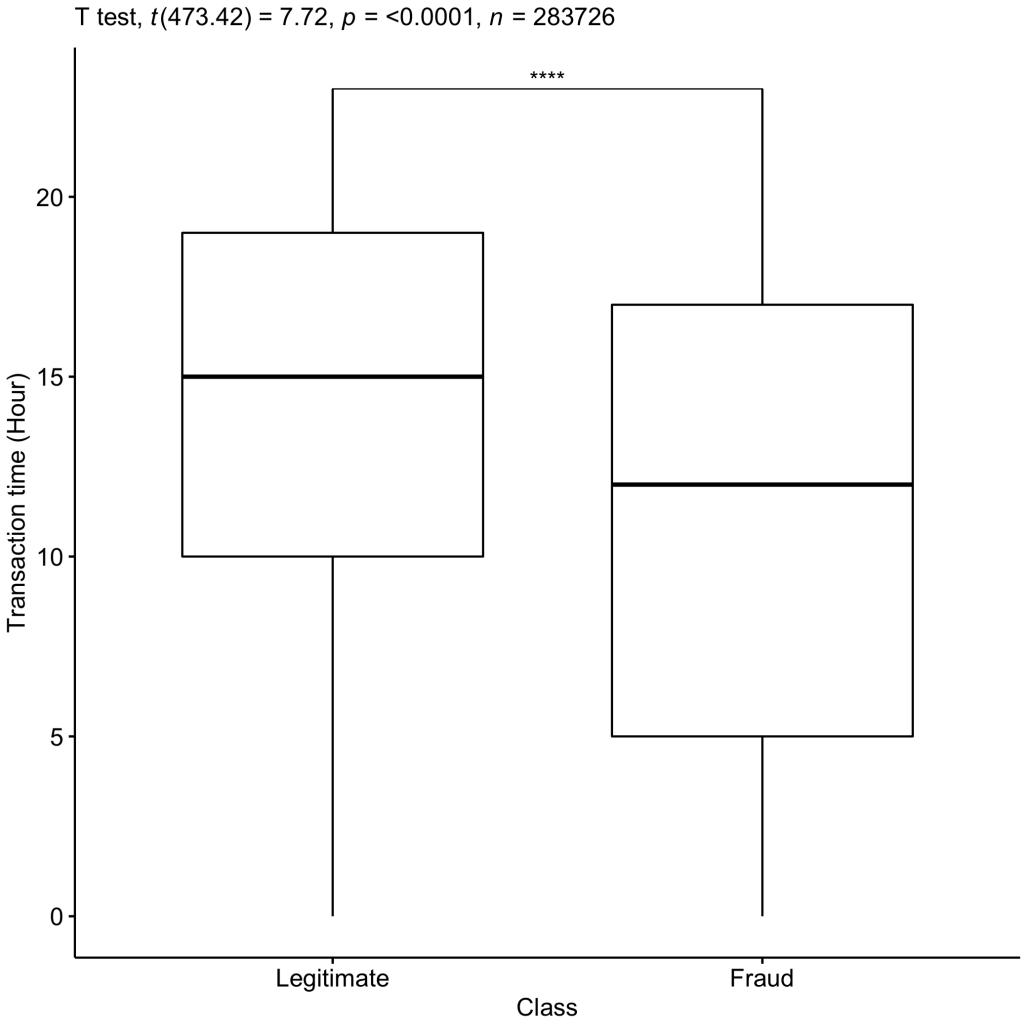


Figure 4: Transaction time by Class

An independent-samples t-test was conducted to compare the transaction time for Legitimate transactions and Fraud transactions (See Figure 4). There were significant differences (t(473.72) = 7.72, p < 0.0001) in the scores with mean score for Legitimate transaction (M = 14.13, SD = 6.23) was higher than Fraud transaction (M = 11.79, SD = 6.57). The magnitude of the differences in the means (mean difference = 2.34, 95% CI: 1.74 to 2.93) was significant. Hence, there is a significant difference in transaction time between Legitimate and Fraud transactions.

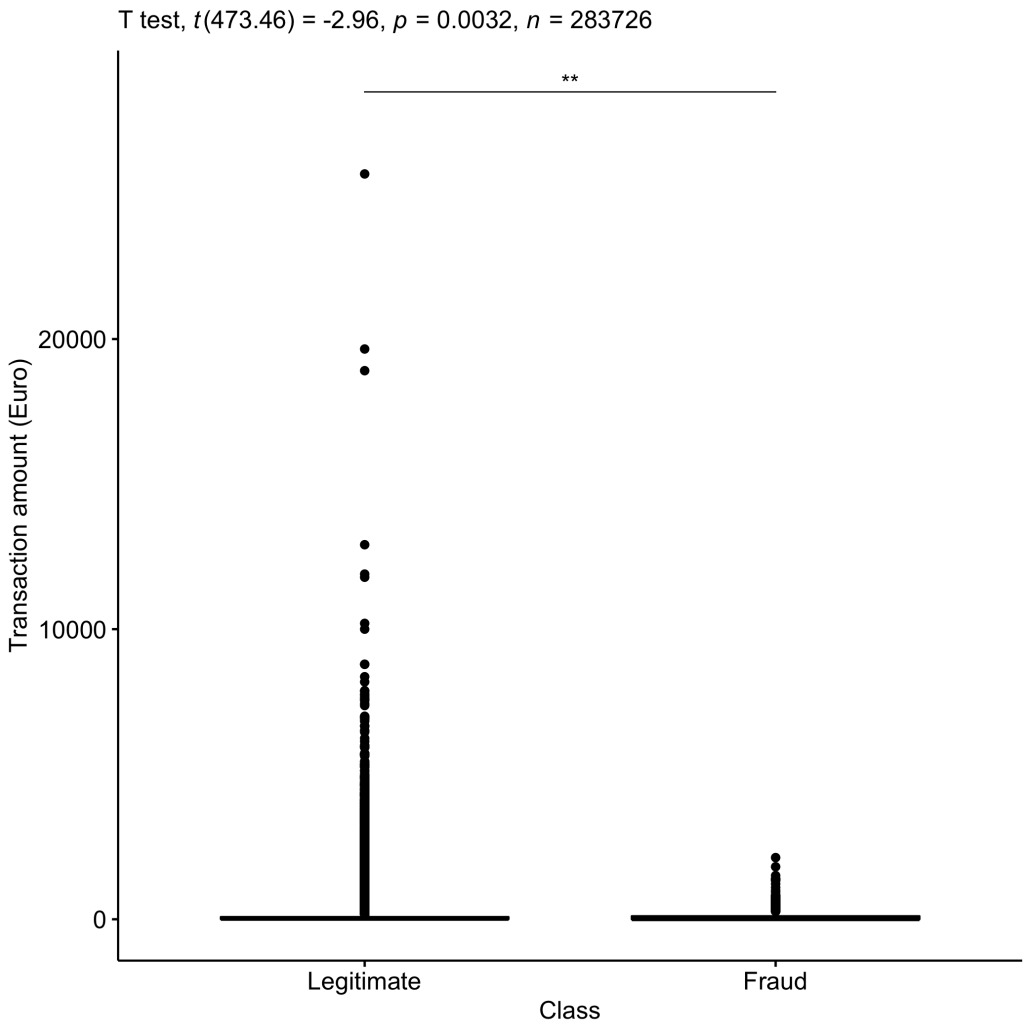


Figure 5: Transaction amount by Class

An independent-samples t-test was conducted to compare the transaction amount for Legitimate transactions and Fraud transactions (See Figure 5). There were significant differences (t(473.46) = -2.96, p = 0.0032) in the scores with mean score for Legitimate transaction (M = 88.41, SD = 250.38) was lower than Fraud transaction (M = 123.87, SD = 260.21). The magnitude of the differences in the means (mean difference = -35.46, 95% CI: -58.99 to -11.93) was significant. Hence, there is a significant difference in transaction amount between Legitimate and Fraud transactions.

3.2 Classification Model

Overall, in this study, Random Forest models perform best, with the mean F_2 score equal 0.8041. Decision Tree models and Logistic Regression models have mean F_2 scores of 0.7877 and 0.6475, respectively. Table 2 shows all three models with their best hyperparameters and mean F_2 scores.

	Mean F_2 Score	Hyperparameters
Logistic Regression	0.6475	C = 100
Decision Tree	0.7877	max_depth = 5
Random Forest	0.8041	max_features = 5 n_estimators = 25

Table 2: Models cross validation

Our final model is a Random Forest Classifier with 25 trees in the forest and 5 features to consider when looking for the best split. Figure 6 shows the confusion matrix when using this model to predict the test set. The model's F_2 score on test set is 0.7987.

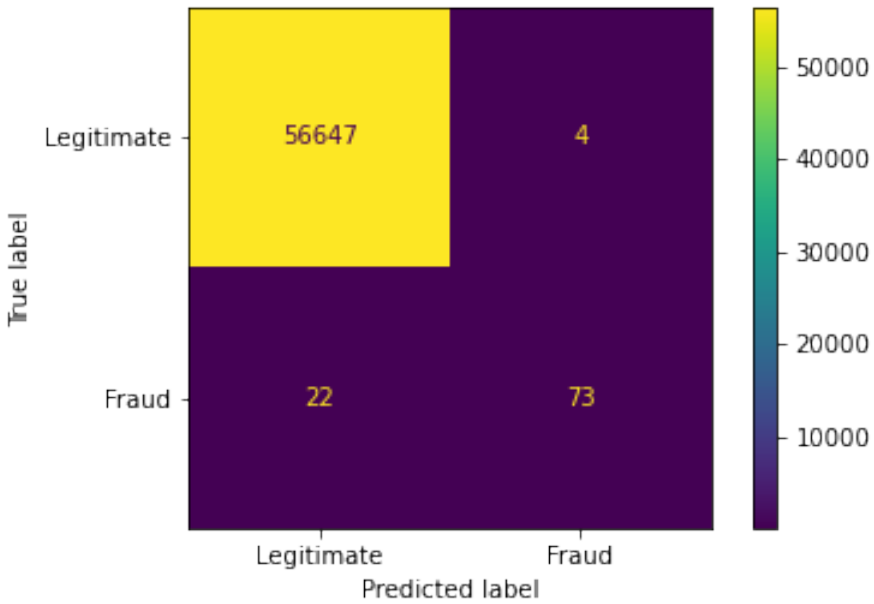


Figure 6: Confusion Matrix

4 Discussion

Nowadays, a solution that minimizes the threat of credit card fraud and is also not too aggressive to block legitimate transactions is needed. Therefore, we emphasize developing an efficient and secure system for detecting fraudsters in our study. We conclude that the model that returns the best results in this dataset is the Random Forest Classifier with 25 trees in the forest and five features to consider when looking for the best split. However, since 28 over 31 variables have already been transformed into principal components, we only have three variables to play with. Therefore, we did not gain much insight from our descriptive analysis. In addition, our method is a naive approach to this problem, so our model may not be good enough to apply in production. Moreover, we have not dealt with the most critical part of this dataset: the imbalance.

References

Bloomenthal, A. (2021). Credit card. *Investopedia*. Retrieved from https://www.investopedia.com/terms/c/creditcard.asp

Fernández, A., García, S., Galar, M., Prati, R., Krawczyk, B., & Herrera, F. (2018). *Learning from imbalanced data sets*. doi: 10.1007/978-3-319-98074-4

He, H., & Ma, Y. (2013). *Imbalanced learning: Foundations, algorithms, and applications* (1st ed.). Wiley-IEEE Press.

Le Borgne, Y.-A., & Bontempi, G. (2021). *Machine learning for credit card fraud detection - practical handbook*. Université Libre de Bruxelles. Retrieved from https://github.com/Fraud-Detection-Handbook/fraud-detection-handbook

Appendices

A Explanatory Analysis notebook

Explanatory Analysis

December 16, 2021

[ ]:
```
library(tidyverse)
library(xtable)
library(ggplot2)
library(ggpubr)
library(ggrepel)
library(gridExtra)
library(rstatix)
```

Import data

[ ]:
```
df = read_csv('datasets/cleaned.csv', show_col_types = FALSE)
df <- df %>% mutate(Class = case_when(Class == 0 ~ "Legitimate",
                                      Class == 1 ~"Fraud"))
df$Class <- factor(df$Class, levels = c("Legitimate", "Fraud"))
```

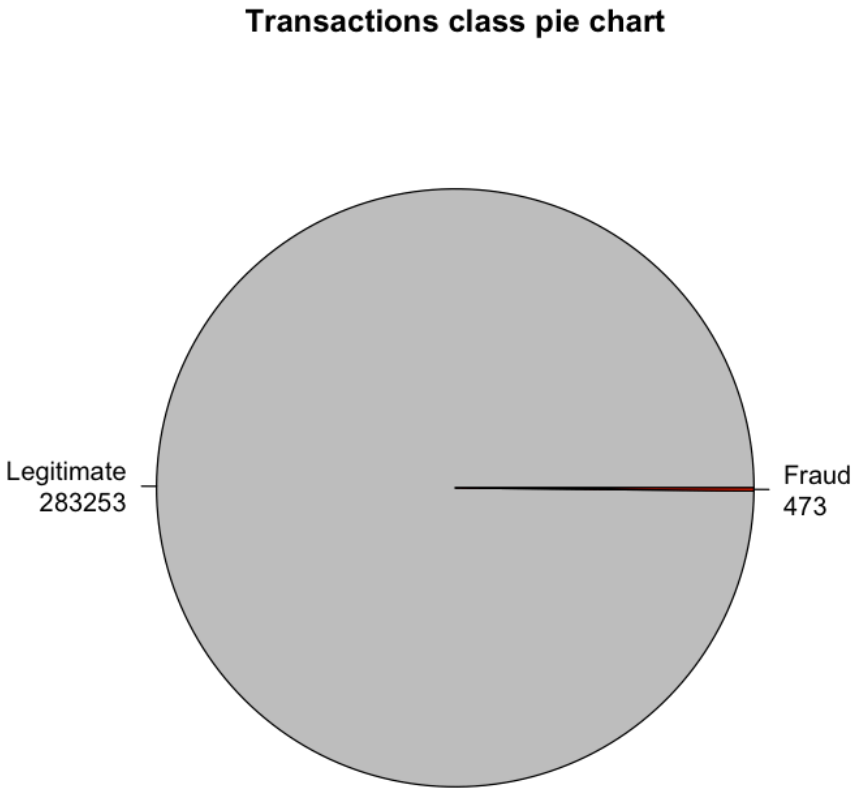
Summary

[3]:
```
df %>% select(c(Hour, Amount, Class)) %>% summary()
```

```
      Hour           Amount                Class
 Min.   : 0.00   Min.   :    0.00   Legitimate:283253
 1st Qu.:10.00   1st Qu.:    5.60   Fraud     :   473
 Median :15.00   Median :   22.00
 Mean   :14.13   Mean   :   88.47
 3rd Qu.:19.00   3rd Qu.:   77.51
 Max.   :23.00   Max.   :25691.16
```

Class pie chart

[4]:
```
t <- table(df$Class)
lbls <- paste(names(t), "\n", t, sep="")
colors <- c("grey", "red")
pie(t, labels = lbls, col = colors,
   main="Transactions class pie chart")
```



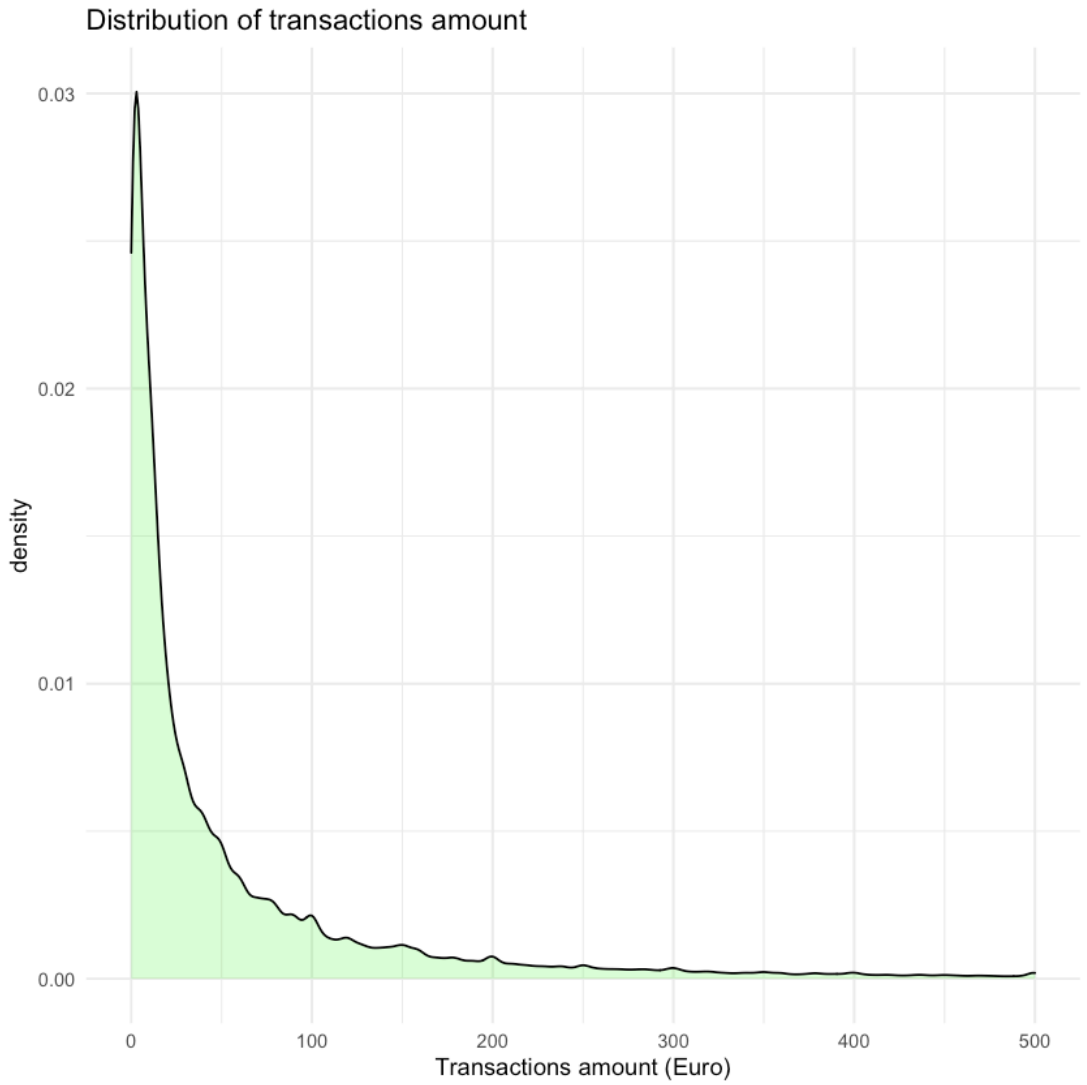
Amount distribution plot

[ ]:
```
g <- ggplot(df, aes(x=Amount))+
  scale_x_continuous()+
  geom_density(fill = "Green", alpha = 0.2)+
  theme_minimal()+
  labs(title = "Distribution of transactions amount")+
  xlim(0,500) + xlab('Transactions amount (Euro)')
```

[6]:
```
g
ggsave("plots/distribution.png", g)
```

```
Warning message:
"Removed 9109 rows containing non-finite values (stat_density)."
```

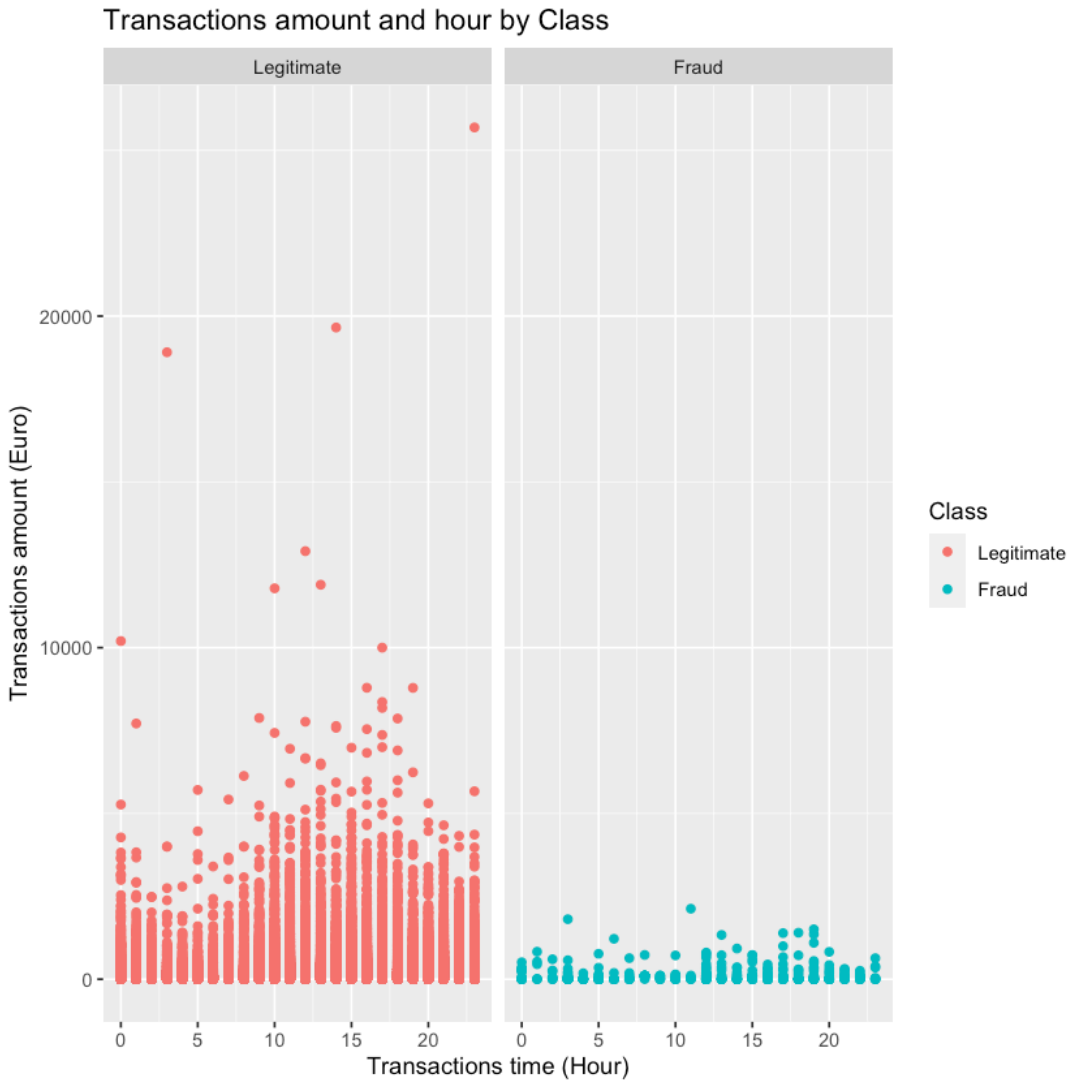
```
Saving 7 x 7 in image

Warning message:
"Removed 9109 rows containing non-finite values (stat_density)."
```



Transactions amount and hour

[7]:
```
g <- ggplot(df,aes(y=Amount,x=Hour,col=Class))+geom_point()+facet_grid(~Class)+
labs(title = "Transactions amount and hour by Class") +
xlab('Transactions time (Hour)') +
ylab('Transactions amount (Euro)')
g
ggsave("plots/scatter.png", g)
```

```
Saving 7 x 7 in image
```



[8]:
```
amount <- df %>% group_by(Class) %>% summarise(mean(Amount), median(Amount),␣
 ↪sd(Amount))
xtable(amount)
```

A xtable: 2 × 4

Class	mean(Amount)	median(Amount)	sd(Amount)
<fct>	<dbl>	<dbl>	<dbl>
Legitimate	88.41357	22.00	250.379
Fraud	123.87186	9.82	260.211

Hour and Class t-test

[16]:
```
df.t_test <- df %>% drop_na(c(Class, Hour))
df.t_test %>%
  group_by(Class) %>%
  get_summary_stats(Hour, type = "mean_sd")
```

A tibble: 2 × 5

Class	variable	n	mean	sd
<fct>	<chr>	<dbl>	<dbl>	<dbl>
Legitimate	Hour	283253	14.129	6.228
Fraud	Hour	473	11.793	6.574

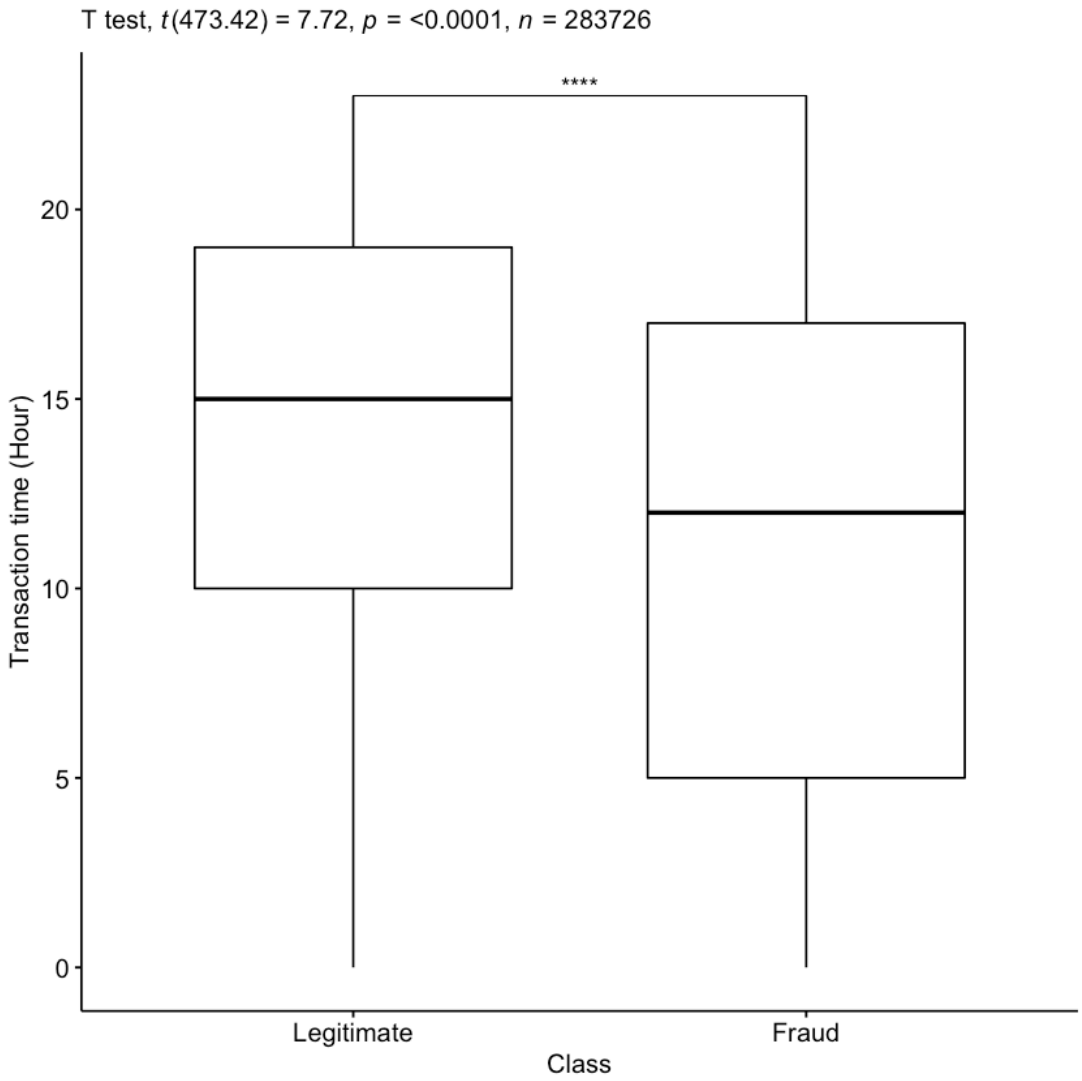
[17]:
```
# Independent sample t-test
stat.test <- df.t_test %>%
             t_test(Hour ~ Class) %>%
             add_significance()
xtable(stat.test)
```

A xtable: 1 × 9

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
<chr>	<chr>	<chr>	<int>	<int>	<dbl>	<dbl>	<dbl>	<chr>
Hour	Legitimate	Fraud	283253	473	7.722633	473.4158	6.85e-14	****

[18]:
```
# Create a box-plot
bxp <- ggboxplot(
  df.t_test, x = "Class", y = "Hour",
  ylab = "Transaction time (Hour)", xlab = "Class")

# Add p-value and significance levels
stat.test <- stat.test %>% add_xy_position(x = "Class")
g <- bxp +
  stat_pvalue_manual(stat.test, tip.length = 0) +
  labs(subtitle = get_test_label(stat.test, detailed = TRUE))
g
ggsave("plots/hour.png", g)
```

```
Saving 7 x 7 in image
```



Amount and Class t-test

[19]:
```
df.t_test <- df %>% drop_na(c(Class, Amount))
df.t_test %>%
  group_by(Class) %>%
  get_summary_stats(Amount, type = "mean_sd")
```

A tibble: 2 × 5

Class	variable	n	mean	sd
<fct>	<chr>	<dbl>	<dbl>	<dbl>
Legitimate	Amount	283253	88.414	250.379
Fraud	Amount	473	123.872	260.211

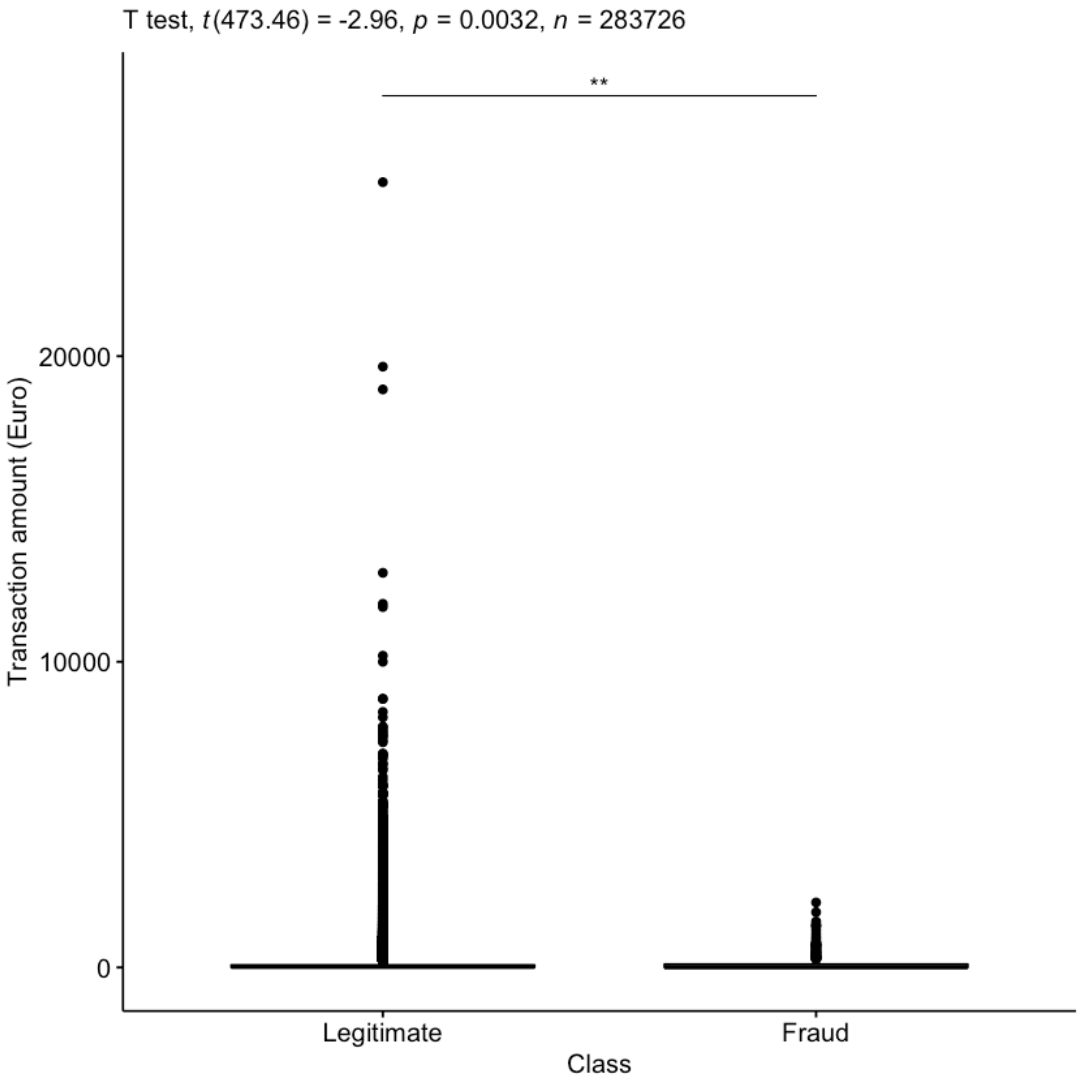
[22]:
```
# Independent sample t-test
stat.test <- df.t_test %>%
             t_test(Amount ~ Class) %>%
             add_significance()
xtable(stat.test)
```

A xtable: 1 × 9

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
<chr>	<chr>	<chr>	<int>	<int>	<dbl>	<dbl>	<dbl>	<chr>
Amount	Legitimate	Fraud	283253	473	-2.961332	473.4606	0.00322	**

[21]:
```
# Create a box-plot
bxp <- ggboxplot(
  df.t_test, x = "Class", y = "Amount",
  ylab = "Transaction amount (Euro)", xlab = "Class")

# Add p-value and significance levels
stat.test <- stat.test %>% add_xy_position(x = "Class")
g <- bxp +
  stat_pvalue_manual(stat.test, tip.length = 0) +
  labs(subtitle = get_test_label(stat.test, detailed = TRUE))
g
ggsave("plots/amount.png", g)
```

```
Saving 7 x 7 in image
```


T test, t(473.46) = -2.96, p = 0.0032, n = 283726
**
20000
10000
0
Transaction amount (Euro)
Legitimate
Fraud
Class

B Machine Learning notebook

Credit Card Fraud Detection

December 16, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
```

[2]:
```
import warnings
warnings.filterwarnings("ignore")
```

1 Data pre-procsessing

[3]:
```
# Import data
df = pd.read_csv('datasets/creditcard.csv')

# Drop NA rows and and duplicate rows
df = df.dropna().drop_duplicates()

# Convert Time variable to Hour
df['Hour'] = df['Time'].apply(datetime.fromtimestamp).dt.hour
df.drop(['Time'], axis=1, inplace=True)
```

[4]:
```
X = df.drop(['Class'], axis=1)
y = df['Class']
```

2 Machine Learning Pipeline

Choose metric for evaluating models

[5]:
```
from sklearn.metrics import fbeta_score, make_scorer, confusion_matrix,␣
↪ConfusionMatrixDisplay

f2_scorer = make_scorer(fbeta_score, beta=2)

def metric(y_test, y_pred):
    print('F2 Score: %.4f' % fbeta_score(y_test, y_pred, beta=2))
    cm = confusion_matrix(y_test, y_pred, labels=[0, 1])
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,␣
↪display_labels=['Legitimate', 'Fraud'])
```

```
    return disp
```

Define standard scaler

[6]:
```
from sklearn.preprocessing import StandardScaler

std_slc = StandardScaler()
```

2.1 Find the best model

[7]:
```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold

cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=68)
```

Logistic Regression

[8]:
```
from sklearn.linear_model import LogisticRegression

# define models and parameters
model = LogisticRegression(random_state=68)
c_values = [100, 10, 1.0, 0.1, 0.01]
# define pipeline and grid search
pipeline = Pipeline(steps=[('scaler', std_slc),
                           ('model', model)])
grid = dict(model__C=c_values)
grid_search = GridSearchCV(estimator=pipeline, param_grid=grid, cv=cv,␣
 ↪scoring=f2_scorer)

# find best hyperparameters
grid_result = grid_search.fit(X, y)

# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.647479 using {'model__C': 100}
```

Decision Tree

[9]:
```
from sklearn.tree import DecisionTreeClassifier

# define models and parameters
model = DecisionTreeClassifier(random_state=68)
max_depth = [2,3,4,5,6,7,8,9,10,20]
# define pipeline and grid search
pipeline = Pipeline(steps=[('scaler', std_slc),
                           ('model', model)])
```

```
grid = dict(model__max_depth=max_depth)
grid_search = GridSearchCV(estimator=pipeline, param_grid=grid, cv=cv,␣
 ↪scoring=f2_scorer)

# find best hyperparameters
grid_result = grid_search.fit(X, y)

# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.787658 using {'model__max_depth': 5}
```

Random Forest

[10]:
```
from sklearn.ensemble import RandomForestClassifier

# define models and parameters
model = RandomForestClassifier(random_state=68)
max_features = ['sqrt', 'log2']
n_estimators = [25,50,100]
# define pipeline and grid search
pipeline = Pipeline(steps=[('scaler', std_slc),
                           ('model', model)])
grid = dict(model__max_features=max_features, model__n_estimators=n_estimators)
grid_search = GridSearchCV(estimator=pipeline, param_grid=grid, cv=cv,␣
 ↪scoring=f2_scorer)

# find best hyperparameters
grid_result = grid_search.fit(X, y)

# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.804068 using {'model__max_features': 'log2', 'model__n_estimators': 25}
```

2.2 Evaluate the best model as result

Split to train dataset and test dataset

[7]:
```
from sklearn.model_selection import train_test_split

df_train, df_test = train_test_split(df, test_size=0.20, random_state=68,␣
 ↪stratify = df[['Class']])
```

Split to features and label

[8]:
```
# Train set
X_train = df_train.drop('Class', axis = 1)
y_train = df_train['Class']
```

```
print("Fraudulent transactions are %.2f%% of the train set." % (y_train.
 ↪value_counts()[1] * 100 / len(y_train)))

# Test set
X_test = df_test.drop('Class', axis = 1)
y_test = df_test['Class']
```

```
Fraudulent transactions are 0.17% of the train set.
```

Normalize Hour and Amount variables

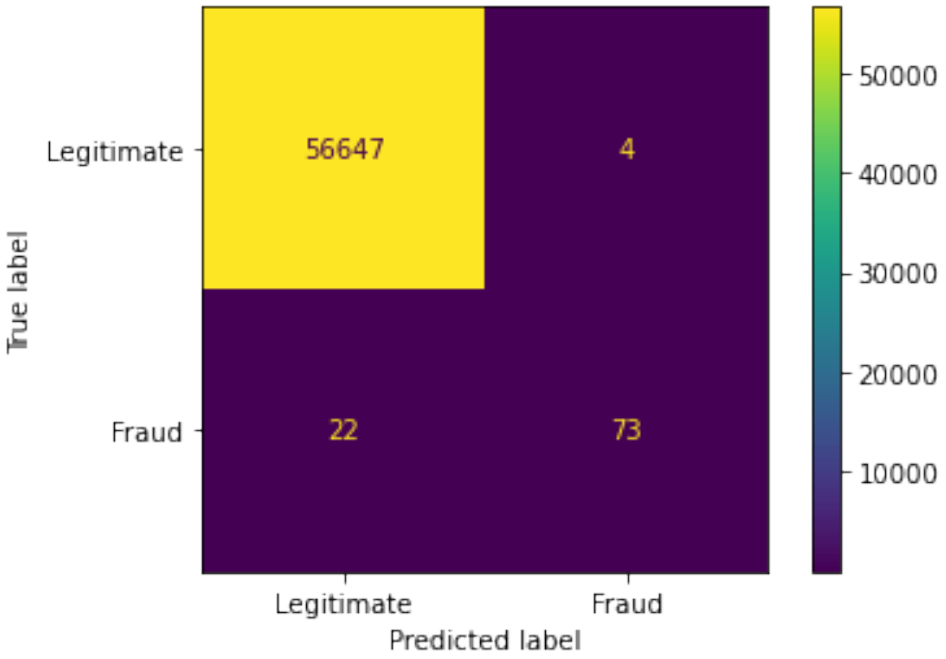
[9]:
```
# Amout variable
scaler_Amount = StandardScaler().fit(X_train['Amount'].values.reshape(-1, 1))
X_train['Amount'] = scaler_Amount.transform(X_train['Amount'].values.
 ↪reshape(-1, 1))
X_test['Amount'] = scaler_Amount.transform(X_test['Amount'].values.reshape(-1,␣
 ↪1))

# Hour variable
scaler_Hour = StandardScaler().fit(X_train['Hour'].values.reshape(-1, 1))
X_train['Hour'] = scaler_Hour.transform(X_train['Hour'].values.reshape(-1, 1))
X_test['Hour'] = scaler_Hour.transform(X_test['Hour'].values.reshape(-1, 1))
```

Evaluate model

[10]:
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(max_features='log2', n_estimators=25,␣
 ↪random_state=68)
y_pred = model.fit(X_train, y_train).predict(X_test)
fig = metric(y_test, y_pred)
```

```
F2 Score: 0.7987
```

[11]:
```
fig.plot()
plt.savefig('plots/confusion_matrix.png')
```


Legitimate
Fraud
True label
56647
4
22
73
Legitimate
Fraud
Predicted label
50000
40000
30000
20000
10000