decision_tree_svm_ccFraud

October 13, 2025

1 Credit Card Fraud Detection with Decision Trees and SVM

Estimated time needed: **30** minutes

In this lab you will consolidate your machine learning (ML) modeling skills by using two popular classification models to identify fraudulent credit card transactions. These models are: Decision Tree and Support Vector Machine. You will use a real dataset of credit card transactions to train each of these models. You will then use the trained model to assess if a credit card transaction is fraudulent or not.

1.1 Objectives

After completing this lab you will be able to:

- Perform basic data preprocessing in Python
- Model a classification task using the Scikit-Learn Python APIs
- Train Suppport Vector Machine and Decision Tree models using Scikit-Learn
- Run inference and assess the quality of the trained models

```
<h2>Introduction</h2>
<br>Imagine that you work for a financial institution and part of your job is to build a model
<br>
<br>You have access to transactions that occured over a certain period of time. The majority o
<br>
<br>This is a Kaggle dataset. You can find this "Credit Card Fraud Detection" dataset from the
```

To train the model, you can use part of the input dataset, while the remaining data can be utilized to assess the quality of the trained model. First, let's import the necessary libraries and download the dataset.

```
<h2>Import Libraries</h2>
```

First, make sure that the required libraries are available. Run the cell below to ensure that.

[1]:
```
!pip install pandas==2.2.3
!pip install scikit-learn==1.6.0
!pip install matplotlib==3.9.3
```

```
Collecting pandas==2.2.3
  Downloading
pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(89 kB)
```

```
Collecting numpy>=1.26.0 (from pandas==2.2.3)
  Downloading
numpy-2.3.3-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata
(62 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas==2.2.3) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-
packages (from pandas==2.2.3) (2024.2)
Collecting tzdata>=2022.7 (from pandas==2.2.3)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas==2.2.3) (1.17.0)
Downloading
pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.7
MB)
                         12.7/12.7 MB
186.5 MB/s eta 0:00:00
Downloading
numpy-2.3.3-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (16.6
MB)
                         16.6/16.6 MB
183.9 MB/s eta 0:00:00
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: tzdata, numpy, pandas
Successfully installed numpy-2.3.3 pandas-2.2.3 tzdata-2025.2
Collecting scikit-learn==1.6.0
  Downloading scikit_learn-1.6.0-cp312-cp312-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
Requirement already satisfied: numpy>=1.19.5 in /opt/conda/lib/python3.12/site-
packages (from scikit-learn==1.6.0) (2.3.3)
Collecting scipy>=1.6.0 (from scikit-learn==1.6.0)
  Downloading
scipy-1.16.2-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata
(62 kB)
Collecting joblib>=1.2.0 (from scikit-learn==1.6.0)
  Downloading joblib-1.5.2-py3-none-any.whl.metadata (5.6 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn==1.6.0)
  Downloading threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Downloading
scikit_learn-1.6.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(13.1 MB)
                         13.1/13.1 MB
141.3 MB/s eta 0:00:00
Downloading joblib-1.5.2-py3-none-any.whl (308 kB)
Downloading
scipy-1.16.2-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (35.7
MB)
                         35.7/35.7 MB
```

```
122.9 MB/s eta 0:00:00a 0:00:01
Downloading threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
Successfully installed joblib-1.5.2 scikit-learn-1.6.0 scipy-1.16.2
threadpoolctl-3.6.0
Collecting matplotlib==3.9.3
  Downloading matplotlib-3.9.3-cp312-cp312-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib==3.9.3)
  Downloading contourpy-1.3.3-cp312-cp312-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (5.5 kB)
Collecting cycler>=0.10 (from matplotlib==3.9.3)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib==3.9.3)
  Downloading fonttools-4.60.0-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl.metadata (111 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib==3.9.3)
  Downloading kiwisolver-1.4.9-cp312-cp312-
manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (6.3 kB)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-
packages (from matplotlib==3.9.3) (2.3.3)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.12/site-packages (from matplotlib==3.9.3) (24.2)
Collecting pillow>=8 (from matplotlib==3.9.3)
  Downloading pillow-11.3.0-cp312-cp312-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (9.0 kB)
Collecting pyparsing>=2.3.1 (from matplotlib==3.9.3)
  Downloading pyparsing-3.2.5-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.12/site-packages (from matplotlib==3.9.3) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.7->matplotlib==3.9.3) (1.17.0)
Downloading
matplotlib-3.9.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.3
MB)
                         8.3/8.3 MB
82.4 MB/s eta 0:00:00
Downloading
contourpy-1.3.3-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (362
kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.60.0-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl (4.9 MB)
                         4.9/4.9 MB
101.2 MB/s eta 0:00:00
Downloading
```

```
kiwisolver-1.4.9-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (1.5
MB)
                         1.5/1.5 MB
78.3 MB/s eta 0:00:00
Downloading
pillow-11.3.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (6.6
MB)
                         6.6/6.6 MB
157.7 MB/s eta 0:00:00
Downloading pyparsing-3.2.5-py3-none-any.whl (113 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler,
contourpy, matplotlib
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.60.0
kiwisolver-1.4.9 matplotlib-3.9.3 pillow-11.3.0 pyparsing-3.2.5
```

To import the libraries that will be used in this lab, execute the cells below.

[3]:
```
# Import the libraries we need to use in this lab
from __future__ import print_function
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize, StandardScaler
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_auc_score
from sklearn.svm import LinearSVC

import warnings
warnings.filterwarnings('ignore')
```

1.2 Load the dataset

Execute the cell below to load the dataset to the variable `raw_data`. The code will fetch the data set for the URL and load the same to the variable. A snapshot of the dataset will be generated as an output.

[4]:
```
# download the dataset
url= "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
  ↪IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/
  ↪creditcard.csv"

# read the input data
raw_data=pd.read_csv(url)
raw_data
```

[4]:
```
             Time         V1         V2        V3        V4        V5  \
0             0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321
1             0.0   1.191857   0.266151  0.166480  0.448154  0.060018
2             1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198
3             1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309
4             2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193
…             …         …          …         …         …         …
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

              V6        V7        V8        V9  …       V21       V22  \
0       0.462388  0.239599  0.098698  0.363787  … -0.018307  0.277838
1      -0.082361 -0.078803  0.085102 -0.255425  … -0.225775 -0.638672
2       1.800499  0.791461  0.247676 -1.514654  …  0.247998  0.771679
3       1.247203  0.237609  0.377436 -1.387024  … -0.108300  0.005274
4       0.095921  0.592941 -0.270533  0.817739  … -0.009431  0.798278
…            …         …         …         …  …         …
284802 -2.606837 -4.918215  7.305334  1.914428  …  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  …  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  …  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  …  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  …  0.261057  0.643078

             V23       V24       V25       V26       V27       V28  Amount  \
0      -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
1       0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
2       0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3      -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
4      -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
…            …         …         …         …         …         …
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
0           0
1           0
2           0
3           0
4           0
…         …
284802      0
```

```
284803       0
284804       0
284805       0
284806       0

[284807 rows x 31 columns]
```

<h2>Dataset Analysis</h2>

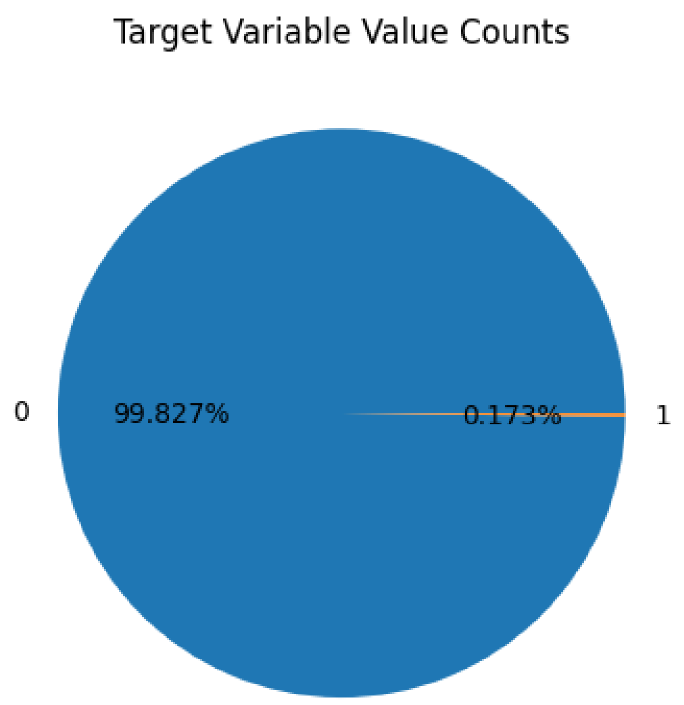
Each row in the dataset represents a credit card transaction. As shown above, each row has 31 variables. One variable (the last variable in the table above) is called Class and represents the target variable. Your objective will be to train a model that uses the other variables to predict the value of the Class variable. Let's first retrieve basic statistics about the target variable.

Note: For confidentiality reasons, the original names of most features are anonymized V1, V2 .. V28. The values of these features are the result of a PCA transformation and are numerical. The feature 'Class' is the target variable and it takes two values: 1 in case of fraud and 0 otherwise. For more information about the dataset please visit this webpage: https://www.kaggle.com/mlg-ulb/creditcardfraud.

[6]:
```python
# get the set of distinct classes
labels = raw_data.Class.unique()

# get the count of each class
sizes = raw_data.Class.value_counts().values

# plot the class value counts
fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, autopct='%1.3f%%')
ax.set_title('Target Variable Value Counts')
plt.show()
```

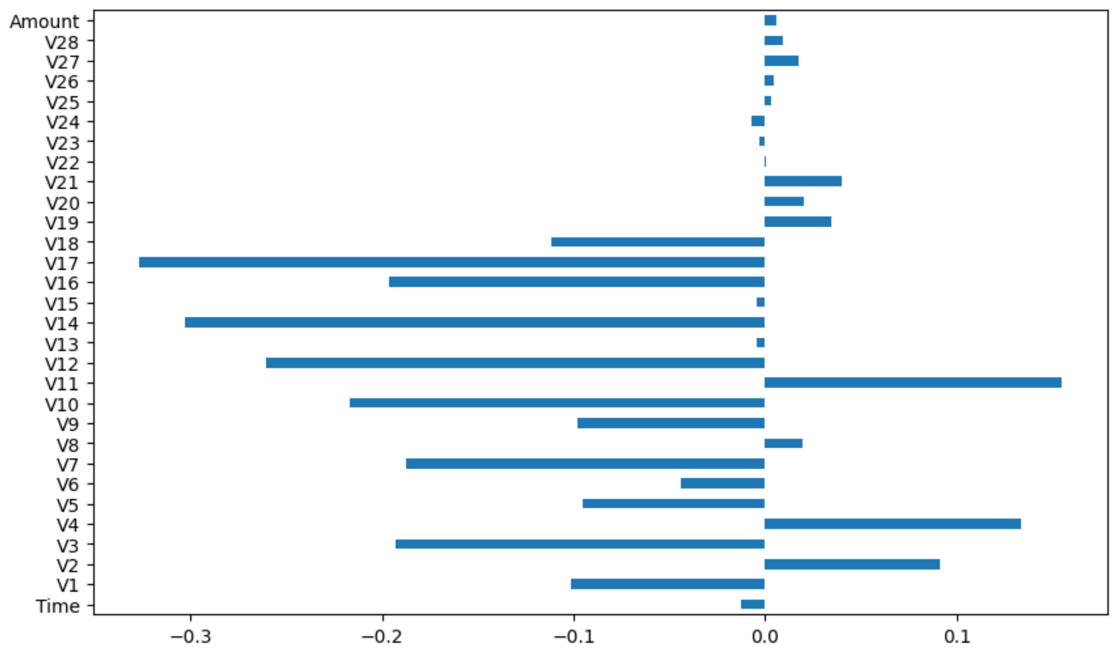


As shown above, the Class variable has two values: 0 (the credit card transaction is legitimate) and 1 (the credit card transaction is fraudulent). Thus, you need to model a binary classification problem. Moreover, the dataset is highly unbalanced, the target variable classes are not represented equally. This case requires special attention when training or when evaluating the quality of a model. One way of handing this case at train time is to bias the model to pay more attention to the samples in the minority class. The models under the current study will be configured to take into account the class weights of the samples at train/fit time.

It is also prudent to understand which features affect the model in what way. We can visualize the effect of the different features on the model using the code below.

```
[7]: correlation_values = raw_data.corr()['Class'].drop('Class')
     correlation_values.plot(kind='barh', figsize=(10, 6))
```

```
[7]: <Axes: >
```



This clearly shows that some features affect the output Class more than the others. For efficient modeling, we may use only the most correlated features.

`<h2>Dataset Preprocessing</h2>`

You will now prepare the data for training. You will apply standard scaling to the input features and normalize them using L_1 norm for the training models to converge quickly. As seen in the data snapshot, there is a parameter called `Time` which we will not be considering for modeling. Hence, features 2 to 30 will be used as input features and feature 31, i.e. Class will be used as the target variable.

[8]:
```
# standardize features by removing the mean and scaling to unit variance
raw_data.iloc[:, 1:30] = StandardScaler().fit_transform(raw_data.iloc[:, 1:30])
data_matrix = raw_data.values

# X: feature matrix (for this analysis, we exclude the Time variable from the␣
 ↪dataset)
X = data_matrix[:, 1:30]

# y: labels vector
y = data_matrix[:, 30]

# data normalization
X = normalize(X, norm="l1")
```

`<h2>Dataset Train/Test Split</h2>`

Now that the dataset is ready for building the classification models, you need to first divide the pre-processed dataset into a subset to be used for training the model (the train set) and a subset to be used for evaluating the quality of the model (the test set).

```
[9]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
      ↪random_state=42)
```

<h2>Build a Decision Tree Classifier model with Scikit-Learn</h2>

Compute the sample weights to be used as input to the train routine so that it takes into account the class imbalance present in this dataset.

```
[10]: w_train = compute_sample_weight('balanced', y_train)
```

Using these sample weights, we may train the Decision Tree classifier. We also make note of the time it takes for training this model to compare it against SVM, later in the lab.

```
[11]: # for reproducible output across multiple function calls, set random_state to a␣
       ↪given integer value
      dt = DecisionTreeClassifier(max_depth=4, random_state=35)

      dt.fit(X_train, y_train, sample_weight=w_train)
```

```
[11]: DecisionTreeClassifier(max_depth=4, random_state=35)
```

<h2>Build a Support Vector Machine model with Scikit-Learn</h2>

Unlike Decision Trees, we do not need to initiate a separate sample_weight for SVMs. We can simply pass a parameter in the scikit-learn function.

```
[12]: # for reproducible output across multiple function calls, set random_state to a␣
       ↪given integer value
      svm = LinearSVC(class_weight='balanced', random_state=31, loss="hinge",␣
       ↪fit_intercept=False)

      svm.fit(X_train, y_train)
```

```
[12]: LinearSVC(class_weight='balanced', fit_intercept=False, loss='hinge',
                random_state=31)
```

<h2>Evaluate the Decision Tree Classifier Models</h2>

Run the following cell to compute the probabilities of the test samples belonging to the class of fraudulent transactions.

```
[13]: y_pred_dt = dt.predict_proba(X_test)[:,1]
```

Using these probabilities, we can evaluate the Area Under the Receiver Operating Characteristic Curve (ROC-AUC) score as a metric of model performance. The AUC-ROC score evaluates your model's ability to distinguish positive and negative classes considering all possible probability thresholds. The higher its value, the better the model is considered for separating the two classes of values.

[14]:
```
roc_auc_dt = roc_auc_score(y_test, y_pred_dt)
print('Decision Tree ROC-AUC score : {0:.3f}'.format(roc_auc_dt))
```

```
Decision Tree ROC-AUC score : 0.939
```

```
<h2>Evaluate the Support Vector Machine Models</h2>
```

Run the following cell to compute the probabilities of the test samples belonging to the class of fraudulent transactions.

[15]:
```
y_pred_svm = svm.decision_function(X_test)
```

You may now evaluate the accuracy of SVM on the test set in terms of the ROC-AUC score.

[16]:
```
roc_auc_svm = roc_auc_score(y_test, y_pred_svm)
print("SVM ROC-AUC score: {0:.3f}".format(roc_auc_svm))
```

```
SVM ROC-AUC score: 0.986
```

1.3 Practice Exercises

Based on what you have learnt in this lab, attempt the following questions.

Q1. Currently, we have used all 30 features of the dataset for training the models. Use the `corr()` function to find the top 6 features of the dataset to train the models on.

[21]:
```
# your code goes here
correlation_values = abs(raw_data.corr()['Class']).drop('Class')
correlation_values = correlation_values.sort_values(ascending=False)[:6]
correlation_values
```

[21]:
```
V17    0.326481
V14    0.302544
V12    0.260593
V10    0.216883
V16    0.196539
V3     0.192961
Name: Class, dtype: float64
```

Click here for solution

```
correlation_values = abs(raw_data.corr()['Class']).drop('Class')
correlation_values = correlation_values.sort_values(ascending=False)[:6]
correlation_values
```

The answer should be ‘V3’,‘V10’,‘V12’,‘V14’,‘V16’ and ‘V17’.

Q2. Using only these 6 features, modify the input variable for training.

[22]:
```
# your code goes here
```

Click here for solution

Replace the statement defining the variable `X` with the following and run the cell again.

```
X = data_matrix[:,[3,10,12,14,16,17]]
```

Q3. Execute the Decision Tree model for this modified input variable. How does the value of ROC-AUC metric change?

Click here for solution

You should observe an increase in the ROC-AUC value with this change for the Decision Tree model.

Q4. Execute the SVM model for this modified input variable. How does the value of ROC-AUC metric change?

Click here for solution

You should observe a decrease in the ROC-AUC value with this change for the SVM model.

Q5. What are the inferences you can draw about Decision Trees and SVMs with what you have learnt in this lab?

Click here for solution

- With a larger set of features, SVM performed relatively better in comparison to the Decision Trees.
- Decision Trees benefited from feature selection and performed better.
- SVMs may require higher feature dimensionality to create an efficient decision hyperplane.

1.3.1 Congratulations! You're ready to move on to your next lesson!

1.4 Author

Abishek Gagneja

Other Contributors

Jeff Grossman

<!– ## Changelog

Date | Version | Changed by | Change Description |

|:————|:——|:—————|:——————————|

2024–1405 | 1.|0 Abhishek Gagnean | Update content and practice exercis|es>

##

[ ]:

```

```