Untitled

March 17, 2023

Import necessary libraries: The first few lines import the necessary libraries for loading and visualizing the dataset, training and evaluating the model, and creating the report.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

Load the dataset: The `pd.read_csv()` function is used to load the dataset from a CSV file and create a `pandas` dataframe.

[8]:
```
# Load the dataset from a CSV file
df = pd.read_csv("dataset.csv")
df.head
```

[8]:
```
<bound method NDFrame.head of      feature_1  feature_2  feature_3  feature_4
feature_5  feature_6  \
0    -1.085631   0.997345   0.282978  -1.506295  -0.578600   1.651437
1    -0.678886  -0.094709   1.491390  -0.638902  -0.443982  -0.434351
2     0.737369   1.490732  -0.935834   1.175829  -1.253881  -0.637752
3    -0.255619  -2.798589  -1.771533  -0.699877   0.927462  -0.173636
4    -0.805367  -1.727669  -0.390900   0.573806   0.338589  -0.011830
..         …          …          …         …          …          …
995  -0.462125   0.357491   0.495822   1.286204  -0.695551  -0.065759
996   0.797611  -0.354865  -1.293321  -0.101811  -0.595784   0.117276
997  -0.585542   0.086091  -0.101158  -0.418401   0.490092   0.703430
998   0.293470   0.652308   0.567850   1.478910  -0.400071  -1.667558
999  -1.060740  -1.285079   1.071255   1.910933   0.649171  -0.597918

     feature_7  feature_8  feature_9  feature_10  target
0    -2.426679  -0.428913   1.265936   -0.866740     1.0
1     2.205930   2.186786   1.004054    0.386186     1.0
2     0.907105  -1.428681  -0.140069   -0.861755     0.0
3     0.002846   0.688223  -0.879536    0.283627     1.0
4     2.392365   0.412912   0.978736    2.238143     0.0
..         …          …          …          …      …
995   0.289966  -0.760412   1.437451   -0.335668     1.0
996  -0.114699   0.745163  -0.635988    0.848133     0.0
```

```
997   1.397362  -1.419400   2.826198    0.793412     1.0
998  -1.350780  -0.112624   0.539086   -0.537706     0.0
999  -1.397932  -0.472266   0.581964    0.970613     1.0

[1000 rows x 11 columns]>
```

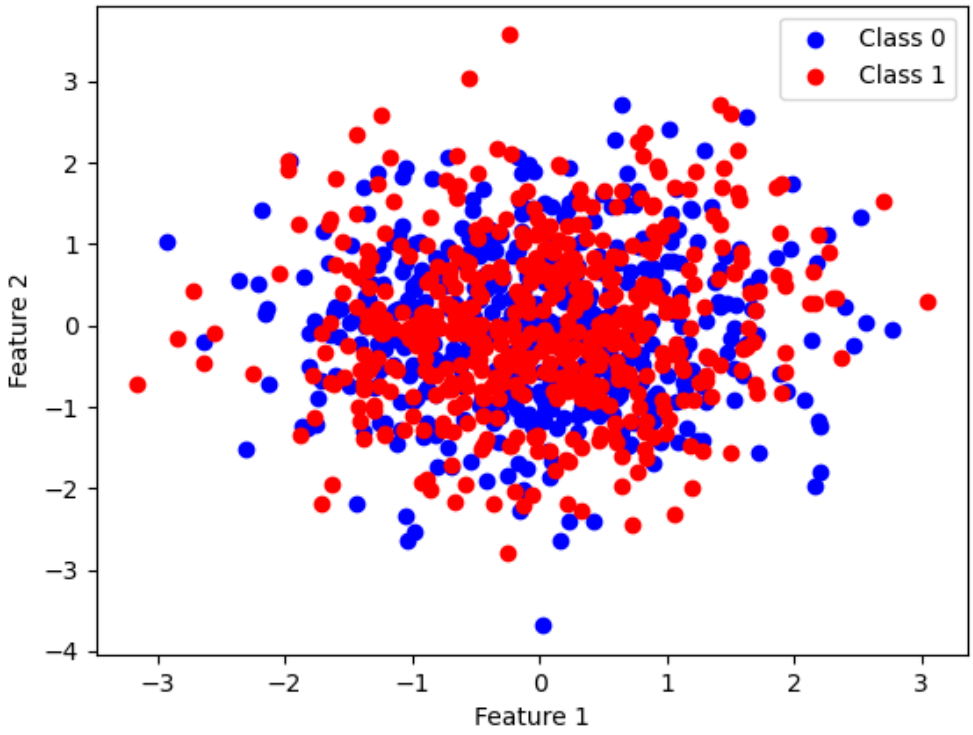
[4]:
```
# Data exploration
print(df.describe())
```

```
         feature_1    feature_2    feature_3    feature_4    feature_5  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean     -0.005297     0.012115     0.017455    -0.011119    -0.008053
std       0.986820     0.998430     0.990776     0.978192     1.003635
min      -3.167055    -3.685499    -3.066988    -3.581474    -3.587494
25%      -0.693351    -0.684767    -0.685417    -0.702782    -0.670098
50%       0.007288    -0.005697    -0.004751     0.052642    -0.025822
75%       0.641805     0.671780     0.713492     0.660364     0.672929
max       3.050755     3.571579     3.386115     2.789487     3.558981

         feature_6    feature_7    feature_8    feature_9   feature_10  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean      0.044696     0.054121    -0.056954     0.028626     0.021528
std       0.986880     1.016036     1.015486     1.023492     0.980629
min      -3.231055    -3.570243    -3.316010    -3.801378    -3.114263
25%      -0.598035    -0.636259    -0.776020    -0.637296    -0.647347
50%       0.082642     0.060037    -0.064200     0.044113     0.035680
75%       0.690880     0.741102     0.618810     0.714136     0.707607
max       3.569280     2.996198     4.068097     3.311735     2.958625

            target
count  1000.000000
mean      0.515000
std       0.500025
min       0.000000
25%       0.000000
50%       1.000000
75%       1.000000
max       1.000000
```

Visualize the dataset: The `plt.scatter()` function is used to create a scatter plot of the dataset, with different colors for each target class. The resulting plot is saved to a PNG file using `plt.savefig()`



Split the data into training and testing sets: The `train_test_split()` function from scikit-learn is used to split the dataset into training and testing sets.

[5]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:, :-1], df.iloc[:,␣
↪-1], test_size=0.2, random_state=123)
```

Train a logistic regression model: The `LogisticRegression()` function is used to create a logistic regression model, which is then trained on the training set using the `fit()` method.

[6]:
```
# Train a logistic regression model
model = LogisticRegression(random_state=123)
model.fit(X_train, y_train)
```

[6]:
```
LogisticRegression(random_state=123)
```

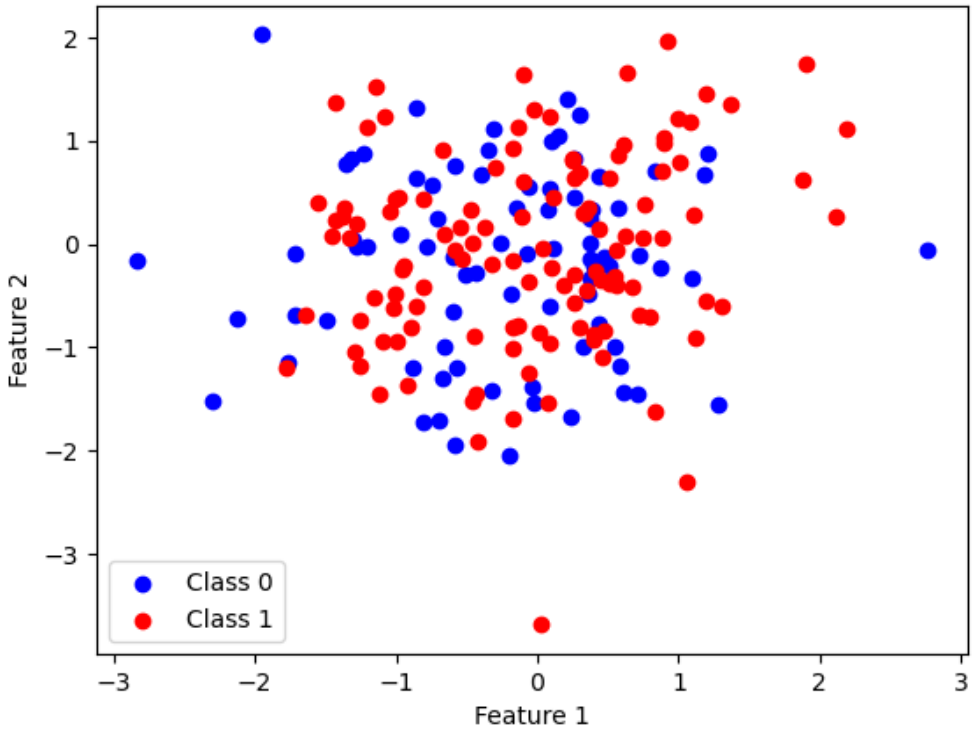
Evaluate the model on the testing set: The trained model is used to make predictions on the testing set using the `predict()` method, and the accuracy of the predictions is calculated using `accuracy_score()` from scikit-learn.

[7]:
```
# Evaluate the model on the testing set
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
# Print the model accuracy
```

```
print(f"Model accuracy: {accuracy:.2f}")
```

```
Model accuracy: 0.50
```

Visualize the model predictions on the testing set: The `plt.scatter()` function is used again to create a scatter plot of the testing set predictions, with different colors for each predicted class. The resulting plot is saved to a PNG file using `plt.savefig()`.




```
[ ]:
```