March 2024

Artificial intelligence

ML Challenge Thomas More
2023-2024

Team 6 - German Novykov, Pidnebesnyi Alex, Mark Vaikul

The challenge

Heart Disease predictions AI

In this challenge we have to design an AI model that will predict heart disease based on symptoms. Ideal use case for this system might be for improving doctors diagnosis.

We were given a training dataset and test dataset. We used three different models to train and then perform predictions. Our result was 91% accuracy on Kaggle.

Models used

Decision Tree

Tree-like graph of decisions and their possible consequences, including chance event outcomes, resource costs, and utility, to arrive at a prediction or decision.

Random forest

Combines the output of multiple decision trees to reach a single result

Linear Regression

Models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data

Machine learning steps

Processing Data

Transform categorical data (gender, location of pain) into numerical. We used OrdinalEncoder for that. For each unique string this encoder creates a number.

Training

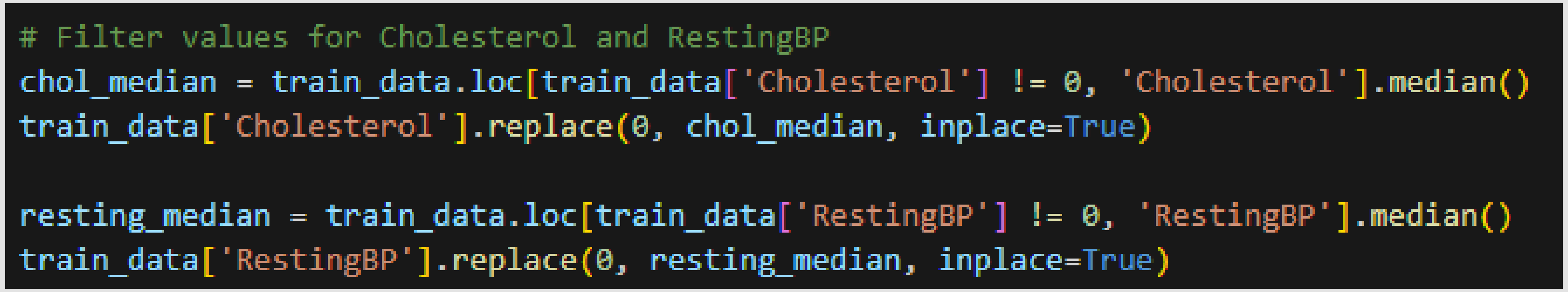
Split the testing data into two datasets - training and testing. We use train split to evaluate our models performance and in effort to not overfit the model.

Testing

Let the model predict the heart disease and save it in the file

Processing Data

Changing faulty data



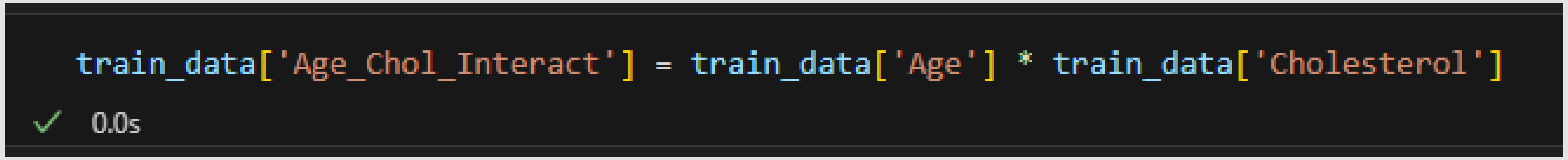
```
# Filter values for Cholesterol and RestingBP
chol_median = train_data.loc[train_data['Cholesterol'] != 0, 'Cholesterol'].median()
train_data['Cholesterol'].replace(0, chol_median, inplace=True)

resting_median = train_data.loc[train_data['RestingBP'] != 0, 'RestingBP'].median()
train_data['RestingBP'].replace(0, resting_median, inplace=True)
```

Faulty data decrease purity of the model. 0 values for Cholesterol and RestingBP may indicate wrongly inserted data

Processing Data

Adding interaction terms



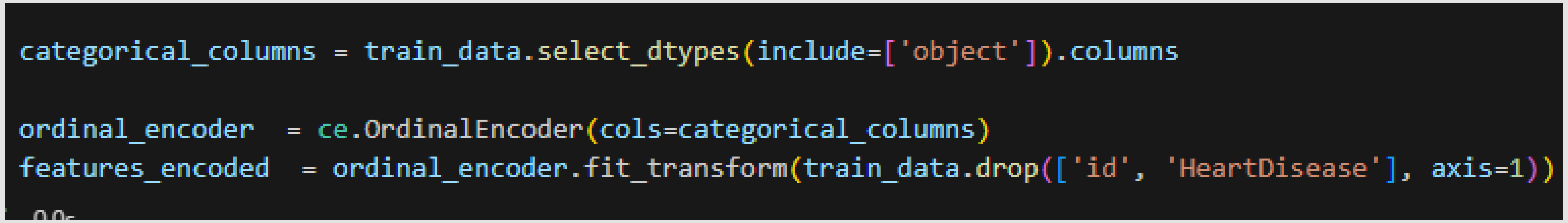
```
train_data['Age_Chol_Interact'] = train_data['Age'] * train_data['Cholesterol']
```

✓ 0.0s

Combining two columns together can increase model accuracy

Processing Data

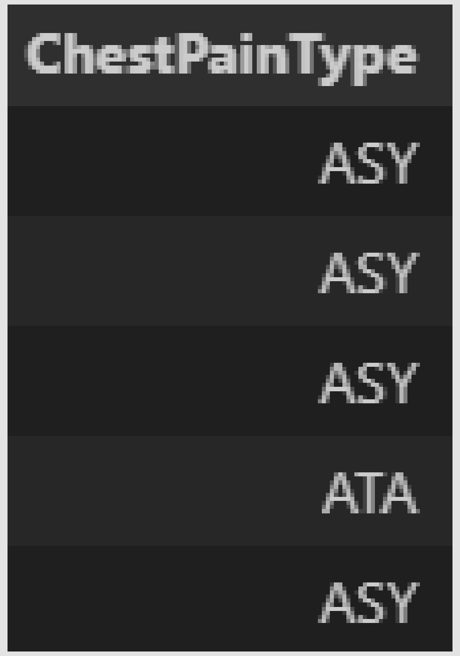
Encoding categorical features



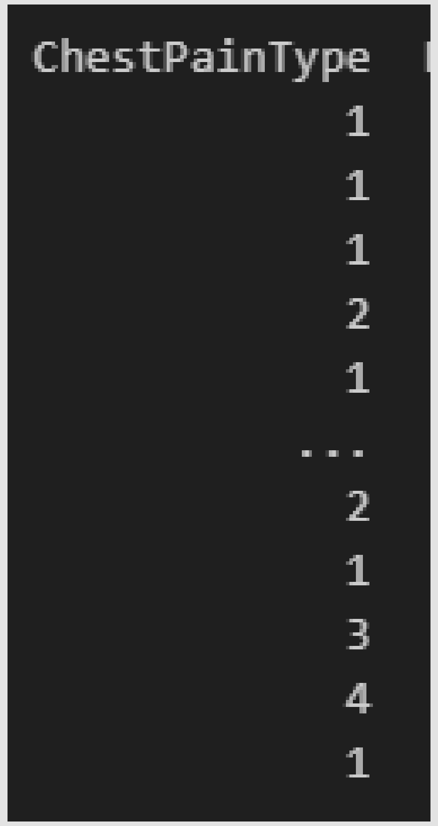
```
categorical_columns = train_data.select_dtypes(include=['object']).columns

ordinal_encoder  = ce.OrdinalEncoder(cols=categorical_columns)
features_encoded  = ordinal_encoder.fit_transform(train_data.drop(['id', 'HeartDisease'], axis=1))
```

Model cannot recognise strings and objects to train on. OrdinalEncoder transforms every string into unique number.



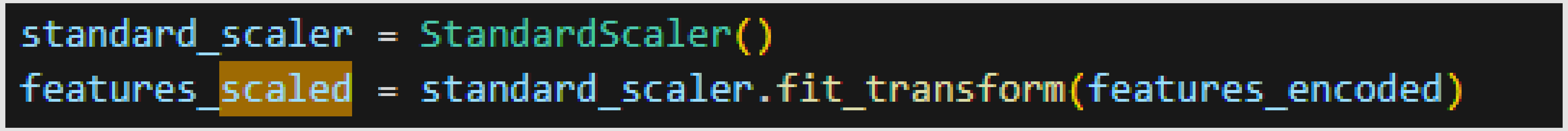
ChestPainType
ASY
ASY
ASY
ATA
ASY



```
ChestPainType
            1
            1
            1
            2
            1
          ...
            2
            1
            3
            4
            1
```

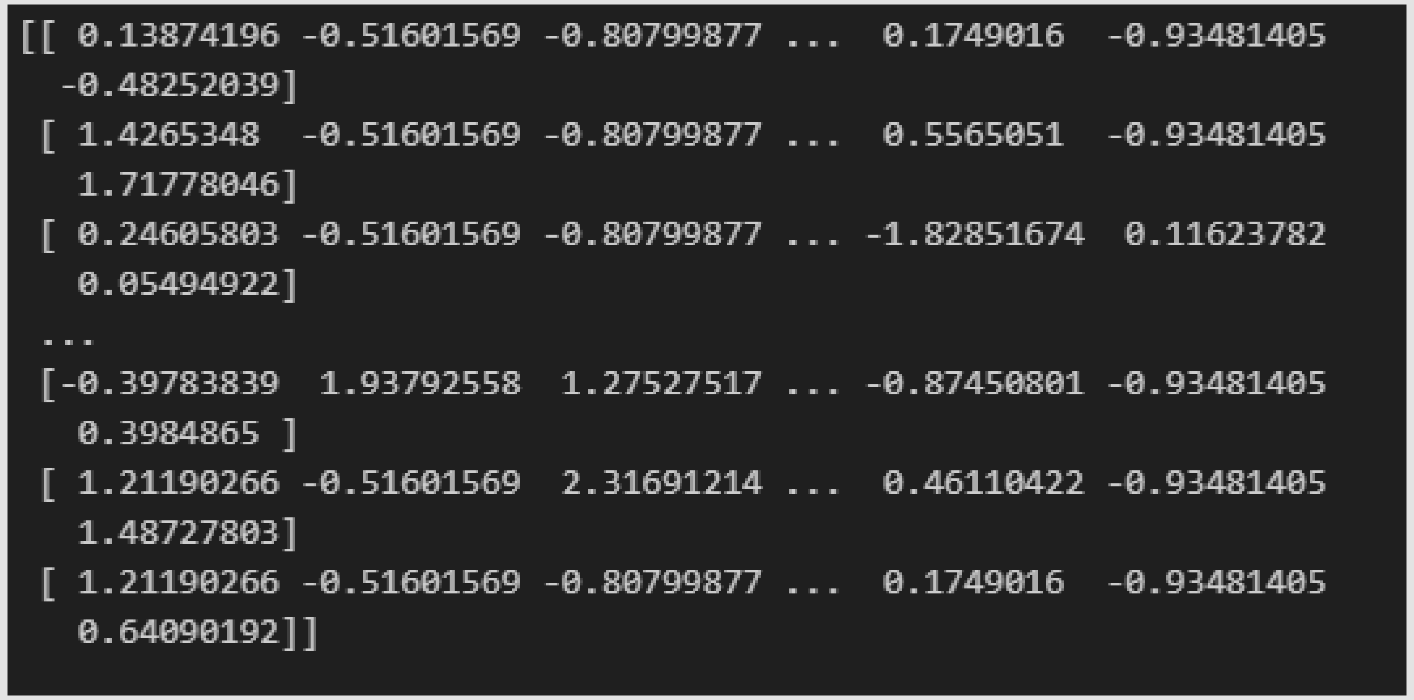
Processing Data

Scaling



```
standard_scaler = StandardScaler()
features_scaled = standard_scaler.fit_transform(features_encoded)
```

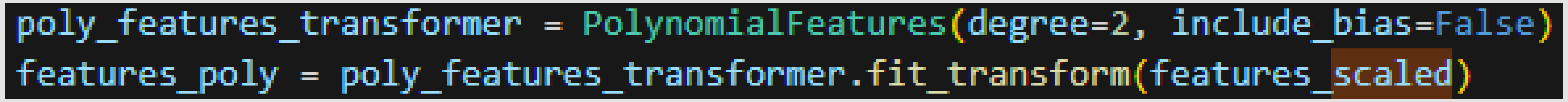
Transforms features to have a mean of 0 and standard deviation of 1. This makes models to converge faster and perform better by treating all features equally in scale.



```
[[ 0.13874196 -0.51601569 -0.80799877 ...  0.1749016  -0.93481405
  -0.48252039]
 [ 1.4265348  -0.51601569 -0.80799877 ...  0.5565051  -0.93481405
   1.71778046]
 [ 0.24605803 -0.51601569 -0.80799877 ... -1.82851674  0.11623782
   0.05494922]
 ...
 [-0.39783839  1.93792558  1.27527517 ... -0.87450801 -0.93481405
   0.3984865 ]
 [ 1.21190266 -0.51601569  2.31691214 ...  0.46110422 -0.93481405
   1.48727803]
 [ 1.21190266 -0.51601569 -0.80799877 ...  0.1749016  -0.93481405
   0.64090192]]
```

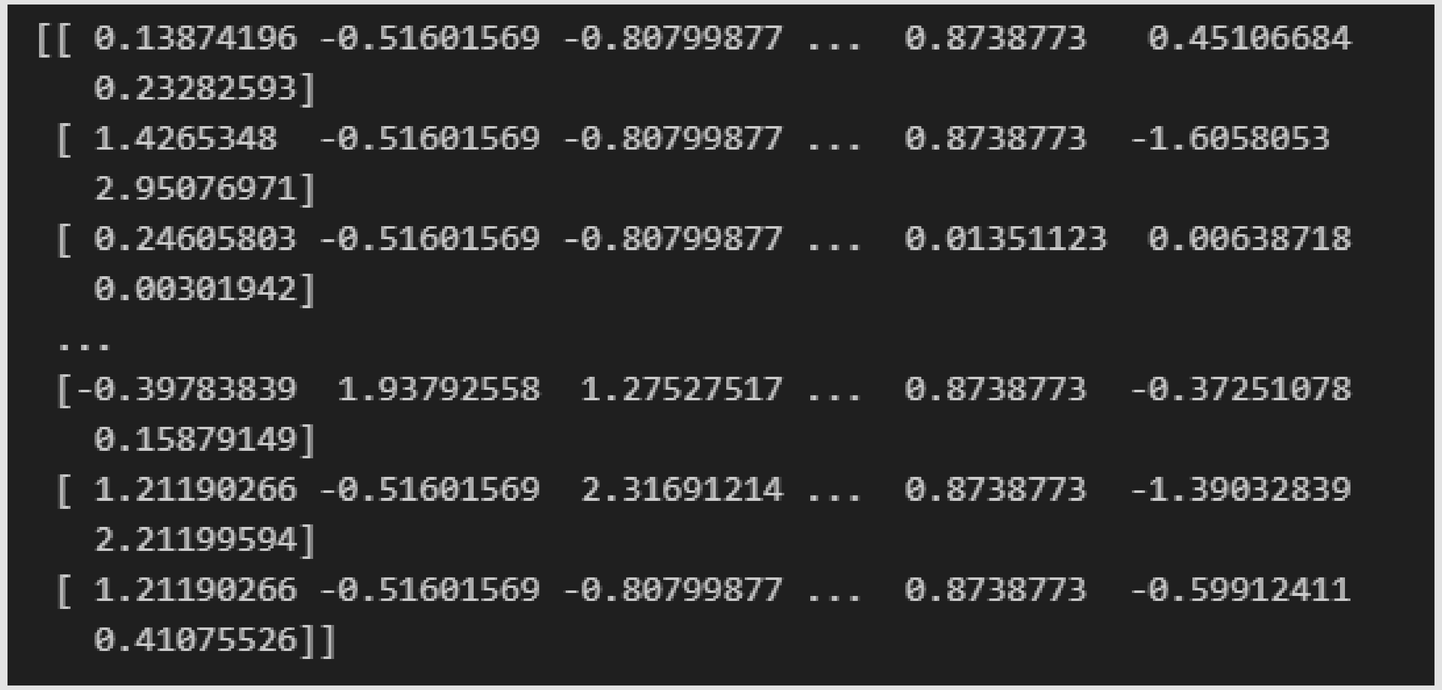
Processing Data

Polynomial Features



```
poly_features_transformer = PolynomialFeatures(degree=2, include_bias=False)
features_poly = poly_features_transformer.fit_transform(features_scaled)
```

For each feature makes a copy with values raised to a degree.

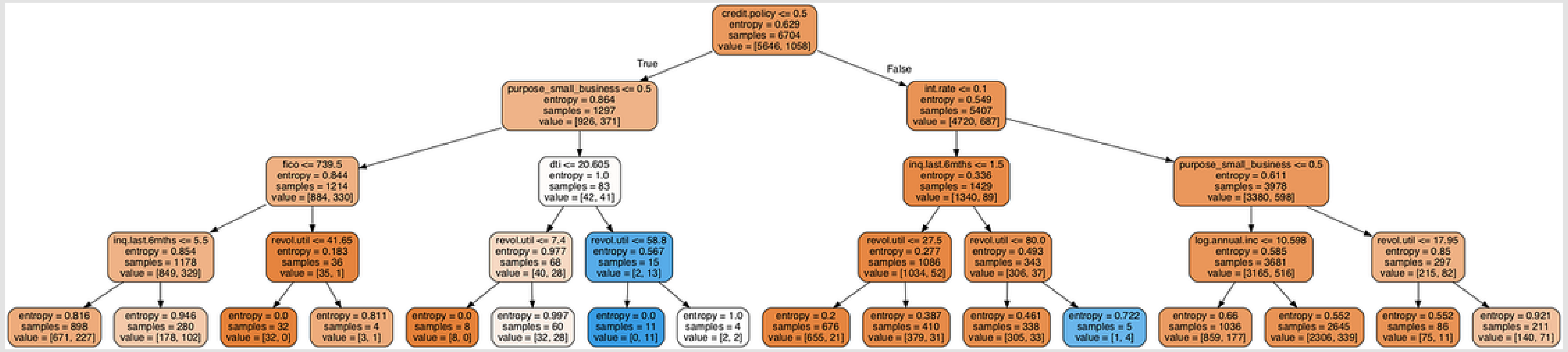


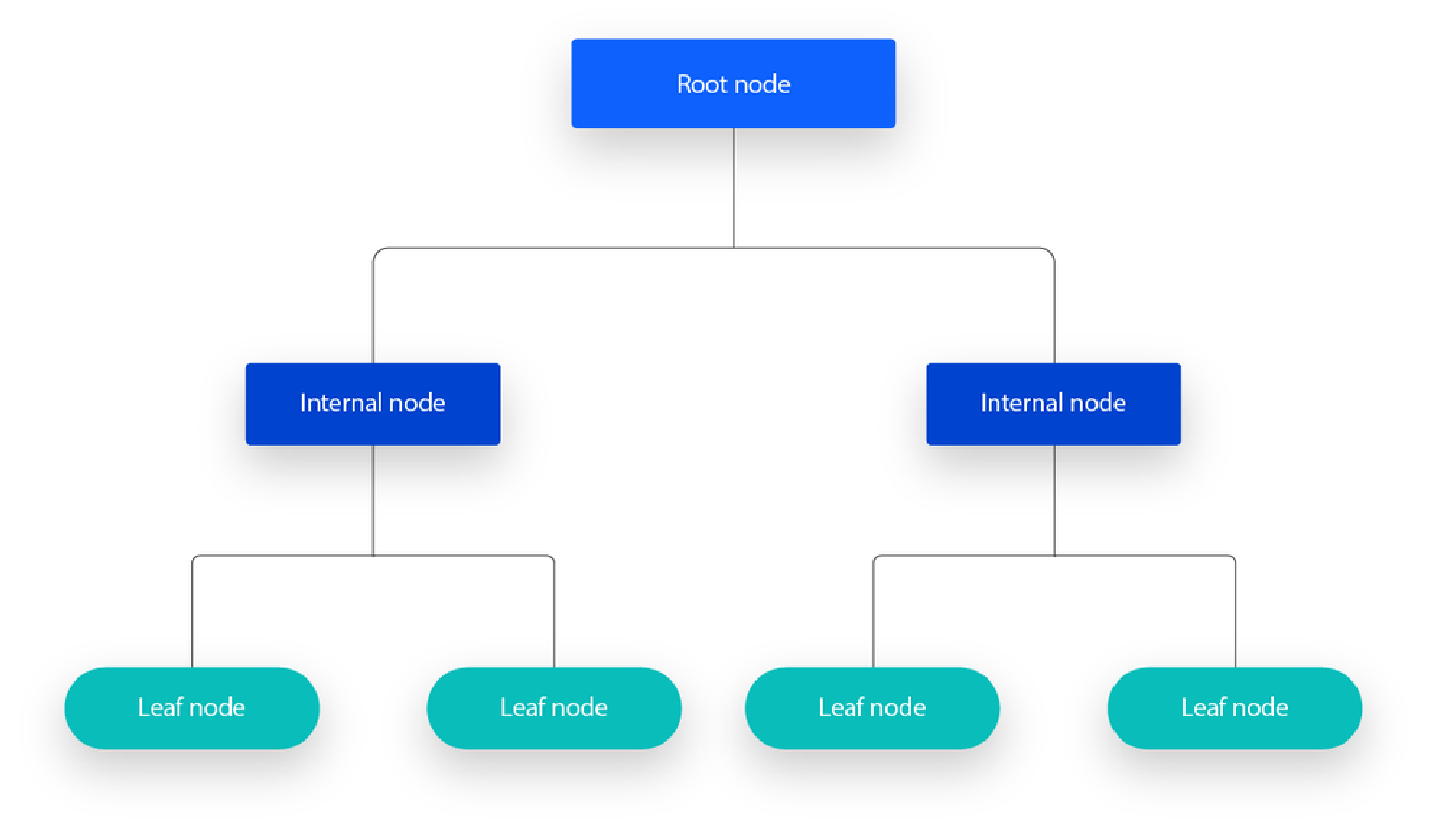
```
[[ 0.13874196 -0.51601569 -0.80799877 ...  0.8738773   0.45106684
   0.23282593]
 [ 1.4265348  -0.51601569 -0.80799877 ...  0.8738773  -1.6058053
   2.95076971]
 [ 0.24605803 -0.51601569 -0.80799877 ...  0.01351123  0.00638718
   0.00301942]
 ...
 [-0.39783839  1.93792558  1.27527517 ...  0.8738773  -0.37251078
   0.15879149]
 [ 1.21190266 -0.51601569  2.31691214 ...  0.8738773  -1.39032839
   2.21199594]
 [ 1.21190266 -0.51601569 -0.80799877 ...  0.8738773  -0.59912411
   0.41075526]]
```

Decision Tree

Decision Tree

Tree structure, which consists of a root node, branches, internal nodes and leaf nodes. Starts with a root node, which does not have any incoming branches. The outgoing branches from the root node then feed into the internal nodes, also known as decision nodes


credit.policy <= 0.5 entropy = 0.629 samples = 6704 value = [5646, 1058]
True
False
purpose_small_business <= 0.5 entropy = 0.864 samples = 1297 value = [926, 371]
int.rate <= 0.1 entropy = 0.549 samples = 5407 value = [4720, 687]
fico <= 739.5 entropy = 0.844 samples = 1214 value = [884, 330]
dti <= 20.605 entropy = 1.0 samples = 83 value = [42, 41]
inq.last.6mths <= 1.5 entropy = 0.336 samples = 1429 value = [1340, 89]
purpose_small_business <= 0.5 entropy = 0.611 samples = 3978 value = [3380, 598]
inq.last.6mths <= 5.5 entropy = 0.854 samples = 1178 value = [849, 329]
revol.util <= 41.65 entropy = 0.183 samples = 36 value = [35, 1]
revol.util <= 7.4 entropy = 0.977 samples = 68 value = [40, 28]
revol.util <= 58.8 entropy = 0.567 samples = 15 value = [2, 13]
revol.util <= 27.5 entropy = 0.277 samples = 1086 value = [1034, 52]
revol.util <= 80.0 entropy = 0.493 samples = 343 value = [306, 37]
log.annual.inc <= 10.598 entropy = 0.585 samples = 3681 value = [3165, 516]
revol.util <= 17.95 entropy = 0.85 samples = 297 value = [215, 82]
entropy = 0.816 samples = 898 value = [671, 227]
entropy = 0.946 samples = 280 value = [178, 102]
entropy = 0.0 samples = 32 value = [32, 0]
entropy = 0.811 samples = 4 value = [3, 1]
entropy = 0.0 samples = 8 value = [8, 0]
entropy = 0.997 samples = 60 value = [32, 28]
entropy = 0.0 samples = 11 value = [0, 11]
entropy = 1.0 samples = 4 value = [2, 2]
entropy = 0.2 samples = 676 value = [655, 21]
entropy = 0.387 samples = 410 value = [379, 31]
entropy = 0.461 samples = 338 value = [305, 33]
entropy = 0.722 samples = 5 value = [1, 4]
entropy = 0.66 samples = 1036 value = [859, 177]
entropy = 0.552 samples = 2645 value = [2306, 339]
entropy = 0.552 samples = 86 value = [75, 11]
entropy = 0.921 samples = 211 value = [140, 71]


Root node

Internal node

Internal node

Leaf node

Leaf node

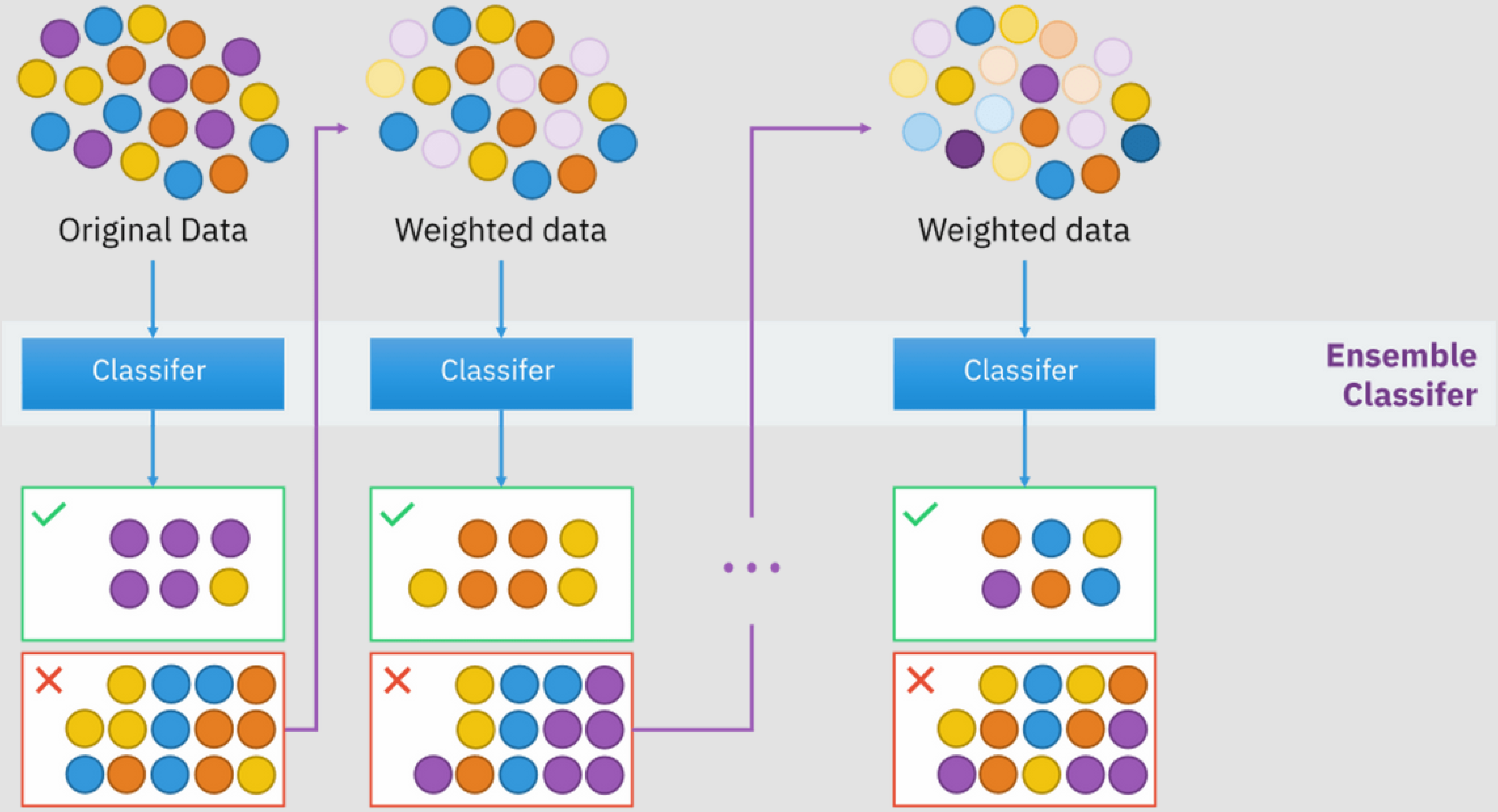
Leaf node

Leaf node

Adaboost

Adaptive Boost

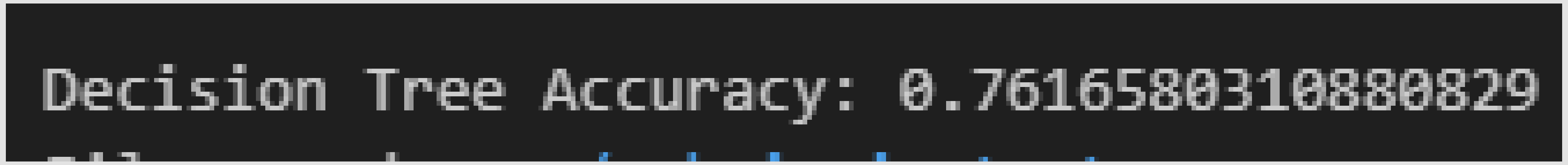
Adaboost works by weighting incorrectly classified instances more heavily so that the subsequent weak learners focus more on the difficult cases.



Decision Tree

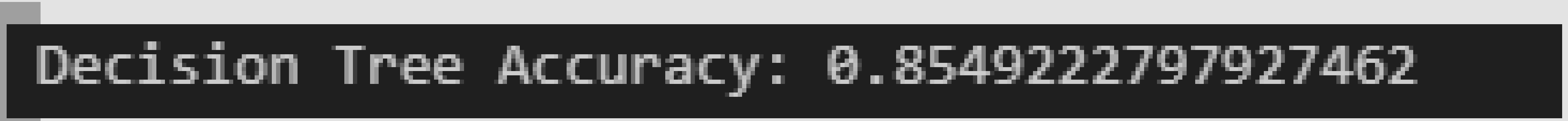
Performance

No hyperparameter tuning



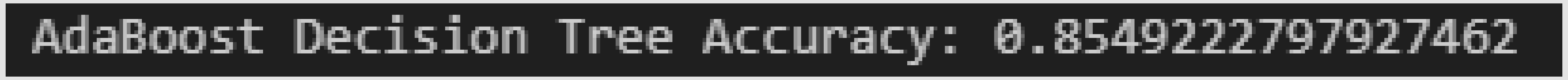
```
Decision Tree Accuracy: 0.7616580310880829
```

Hyperparameter tuning with GridSearch

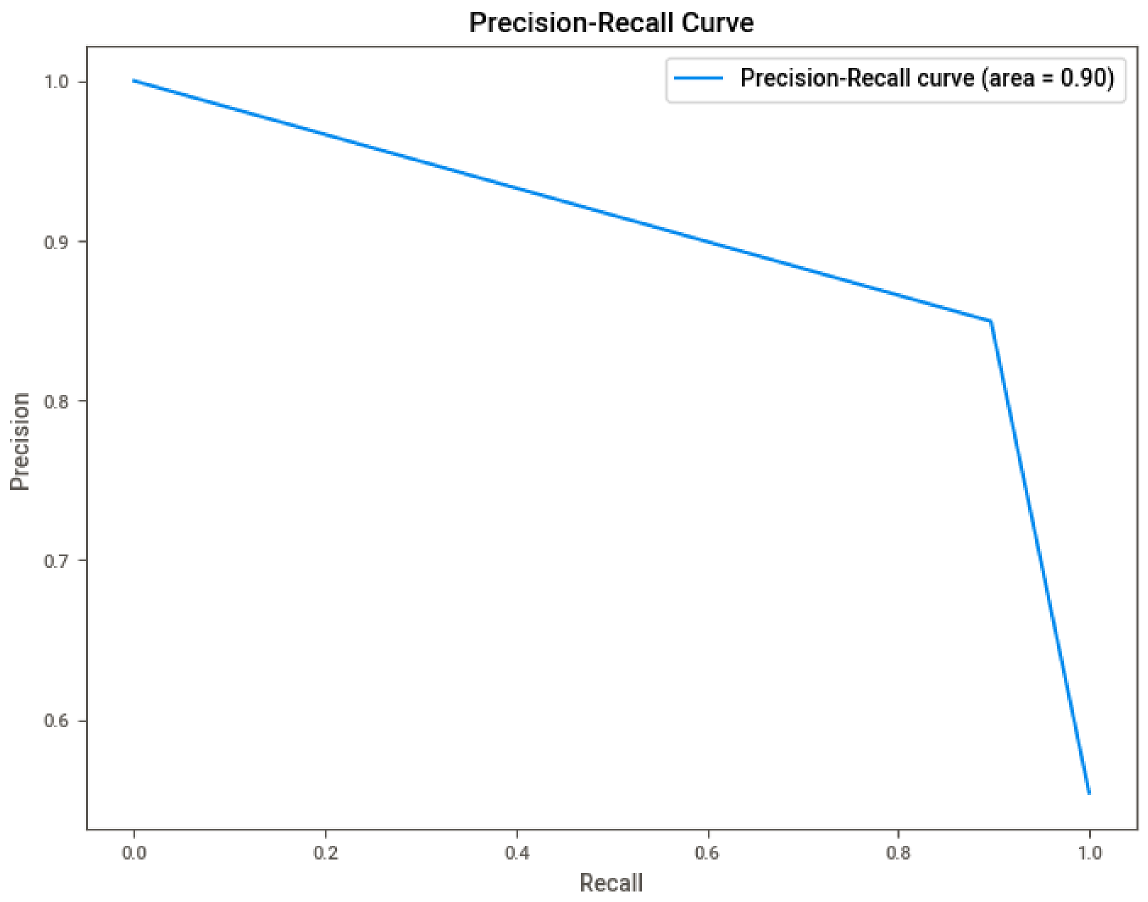


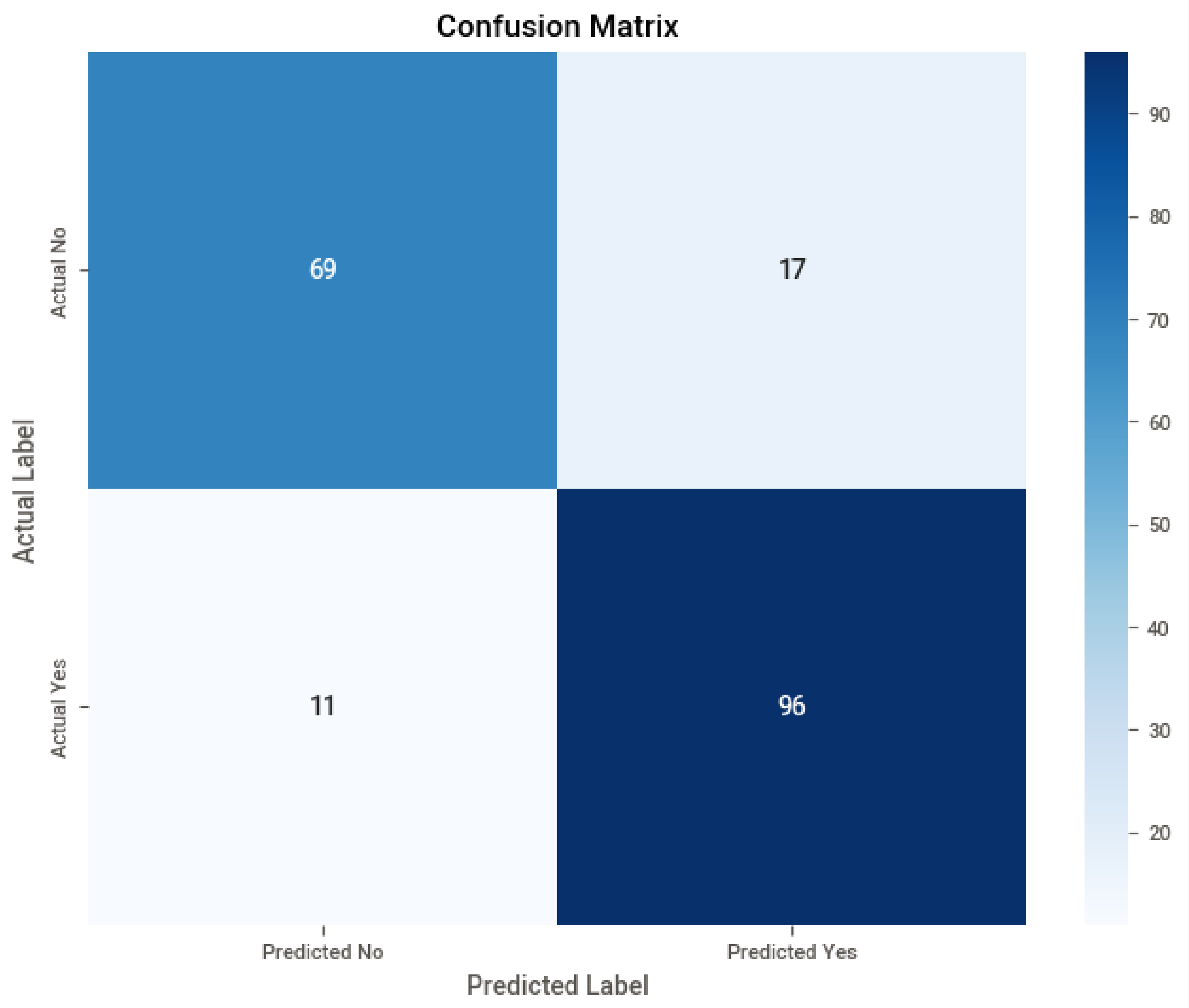
```
Decision Tree Accuracy: 0.8549222797927462
```

With Adaboost



```
AdaBoost Decision Tree Accuracy: 0.8549222797927462
```

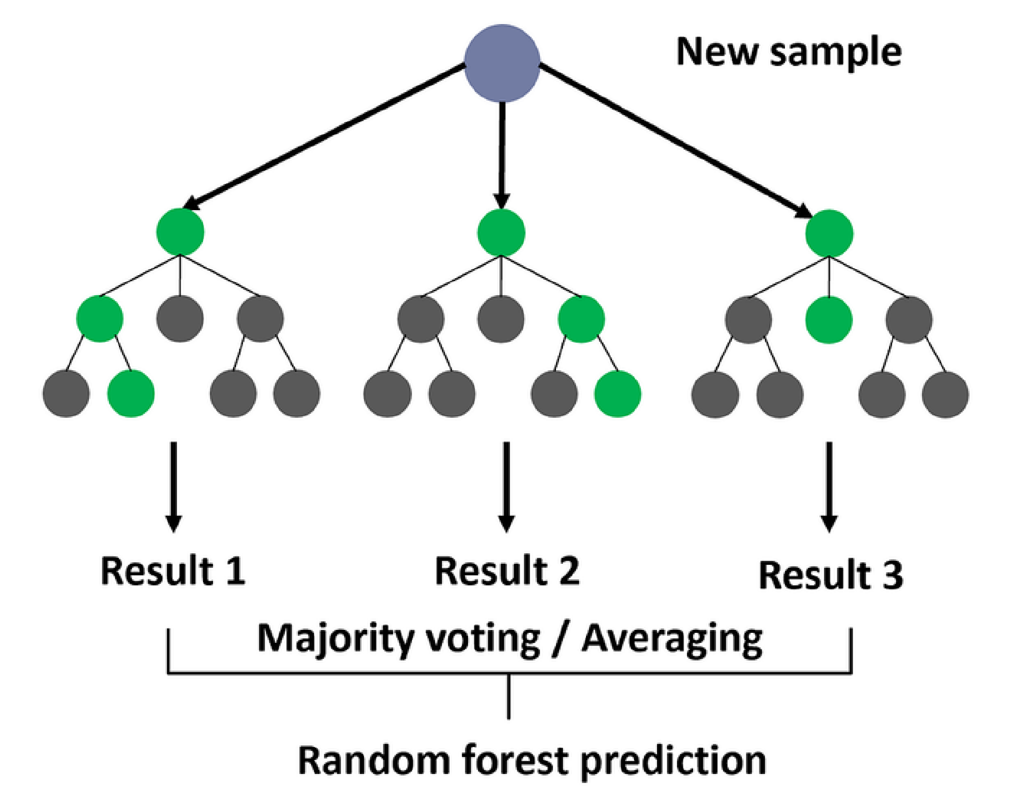




Random Forest

Random forest

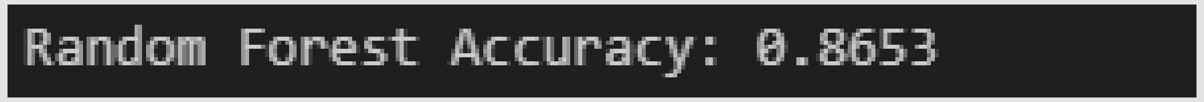
The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees, thus limiting the risk of overfitting



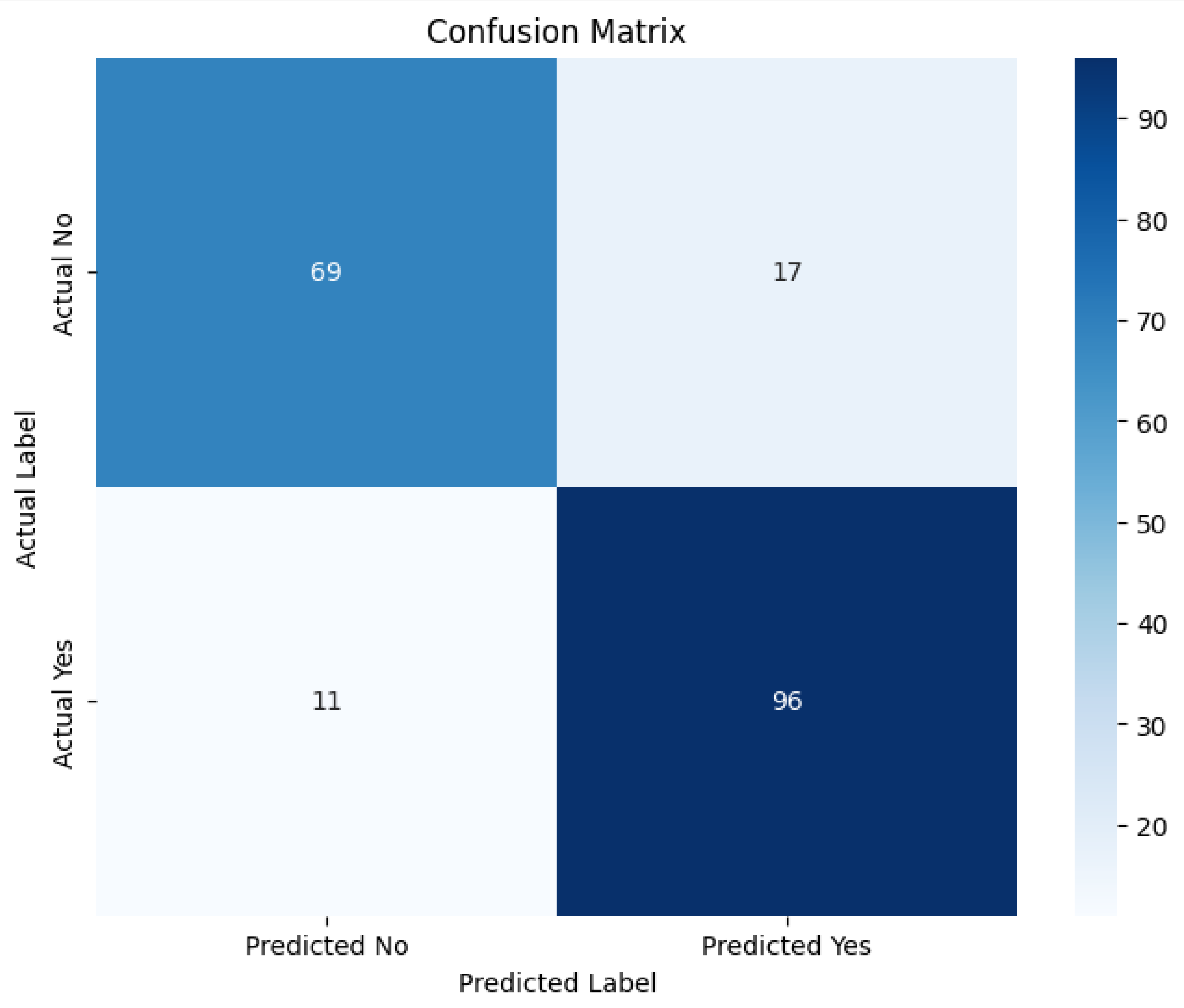
Random Forest

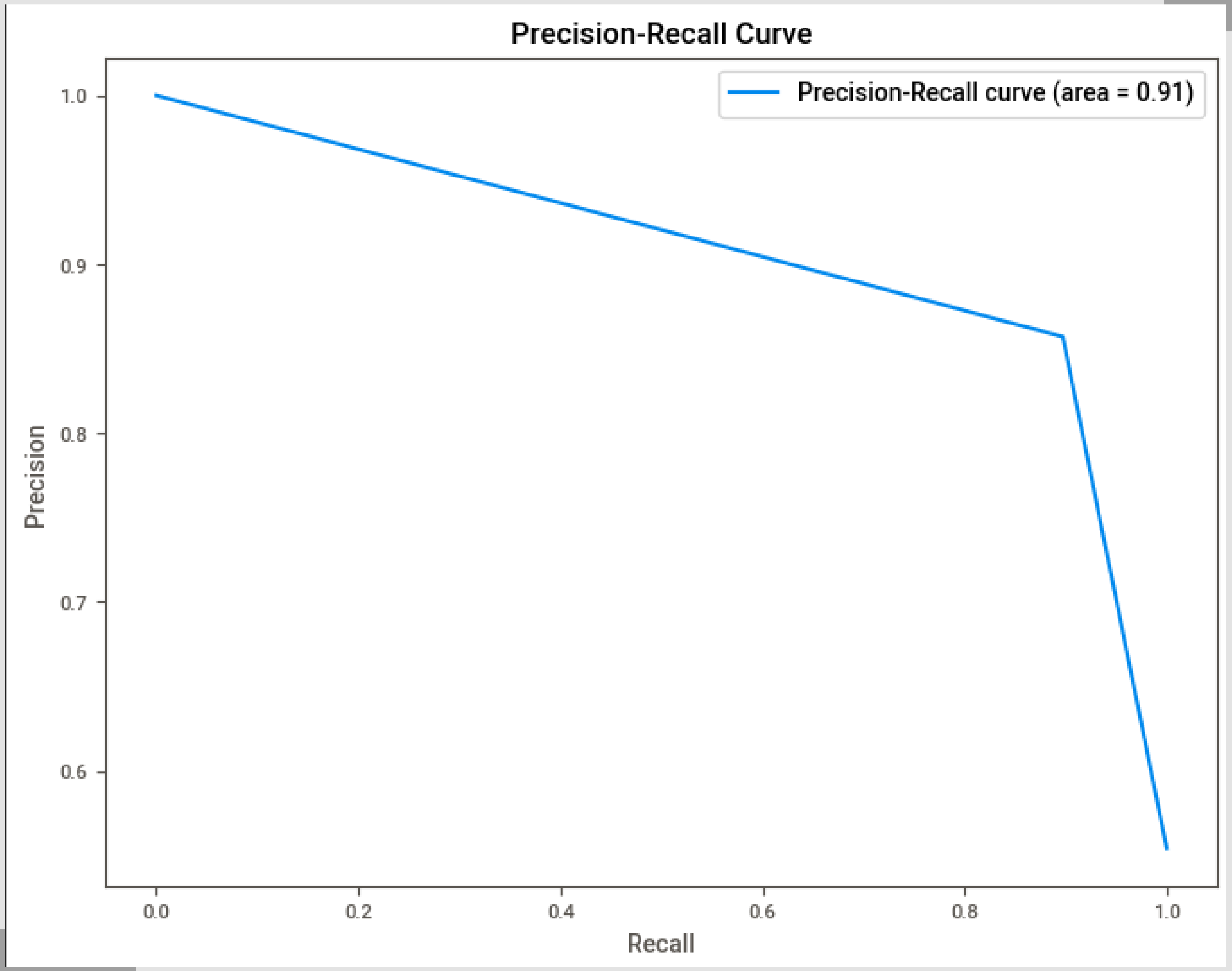
Performance

Random Forest accuracy



```
Random Forest Accuracy: 0.8653
```

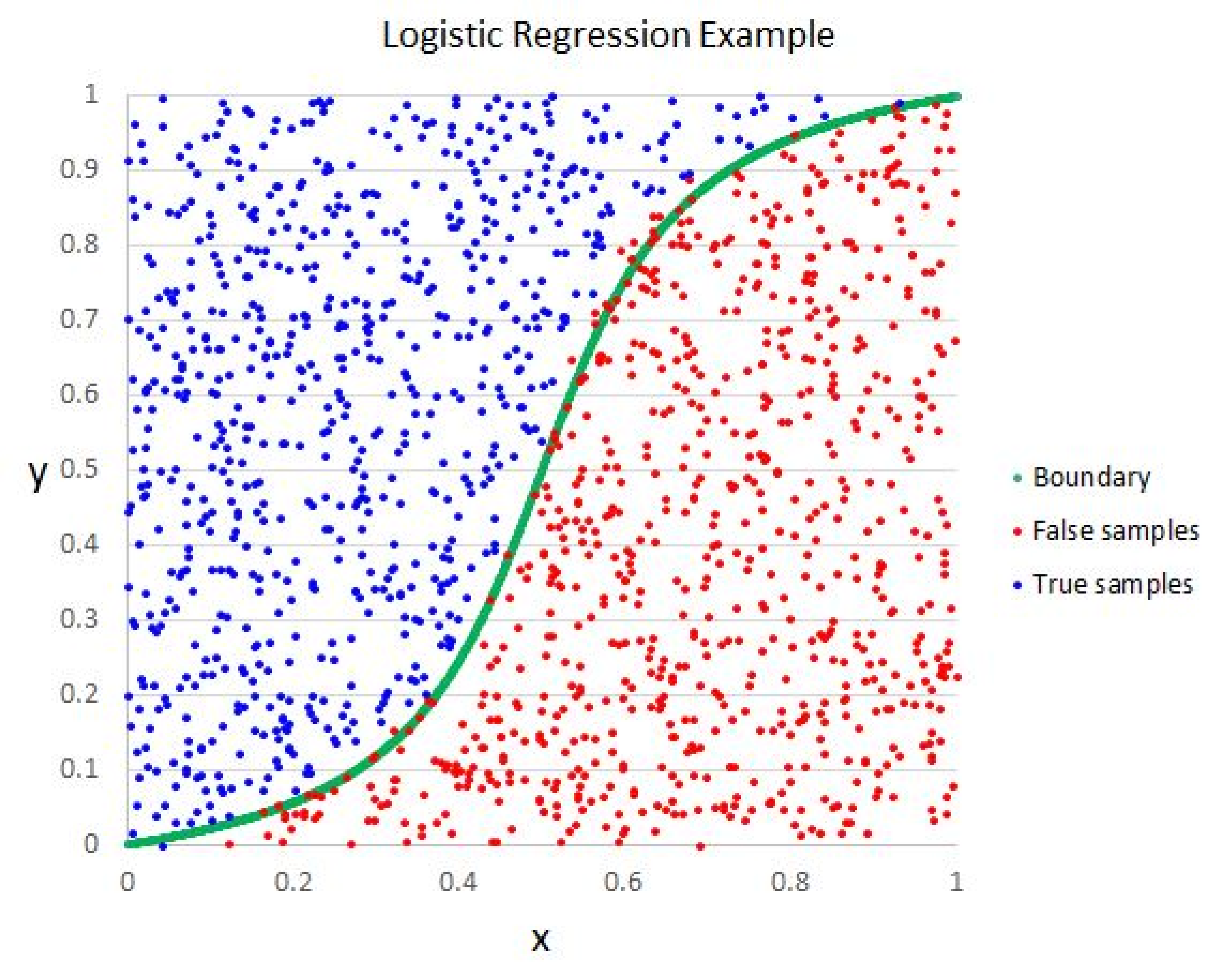




Logistic Regression

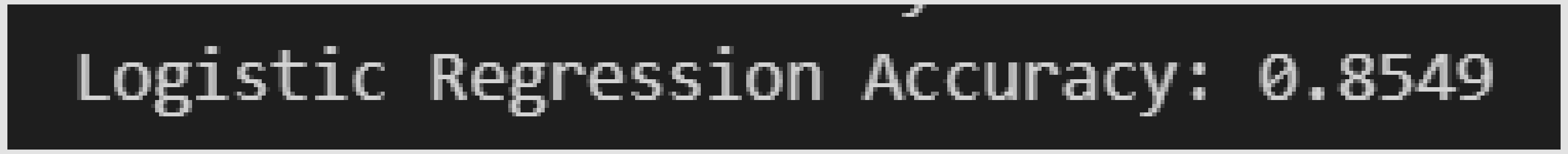
Logistic Regression

Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables.

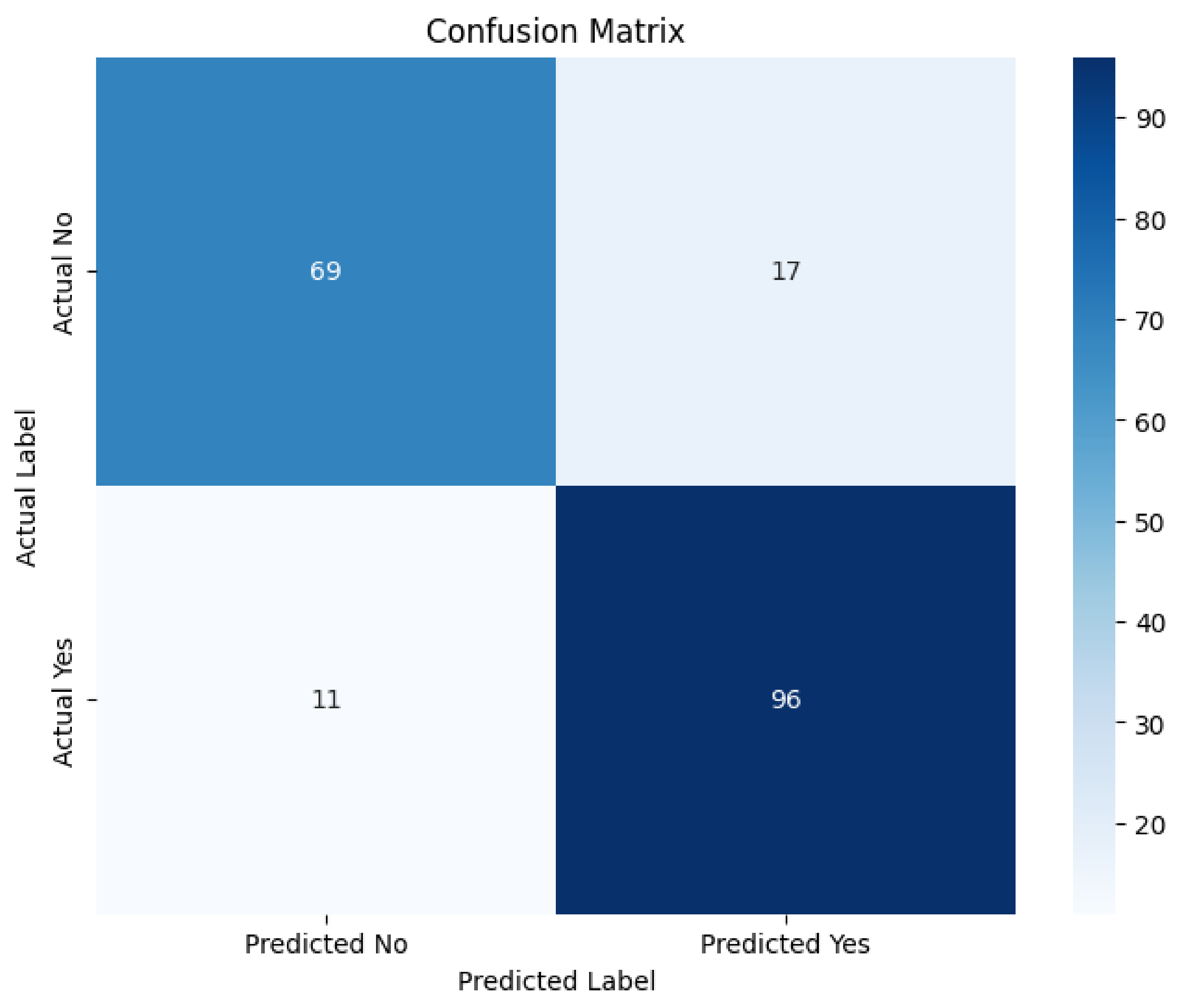


Logistic Regression

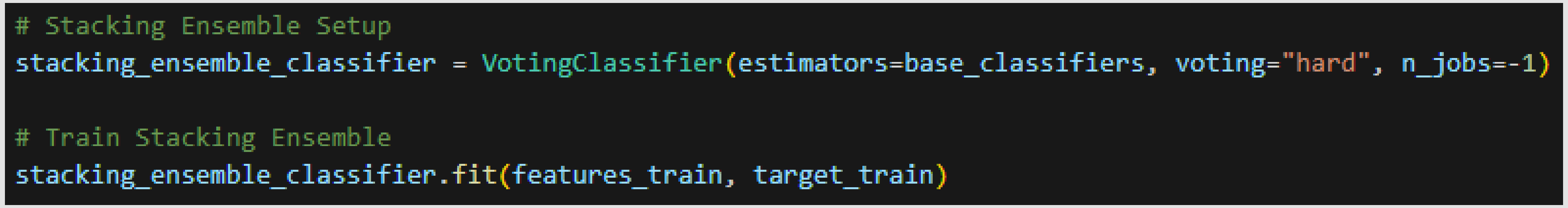
Preformance



```
Logistic Regression Accuracy: 0.8549
```



Combining the models

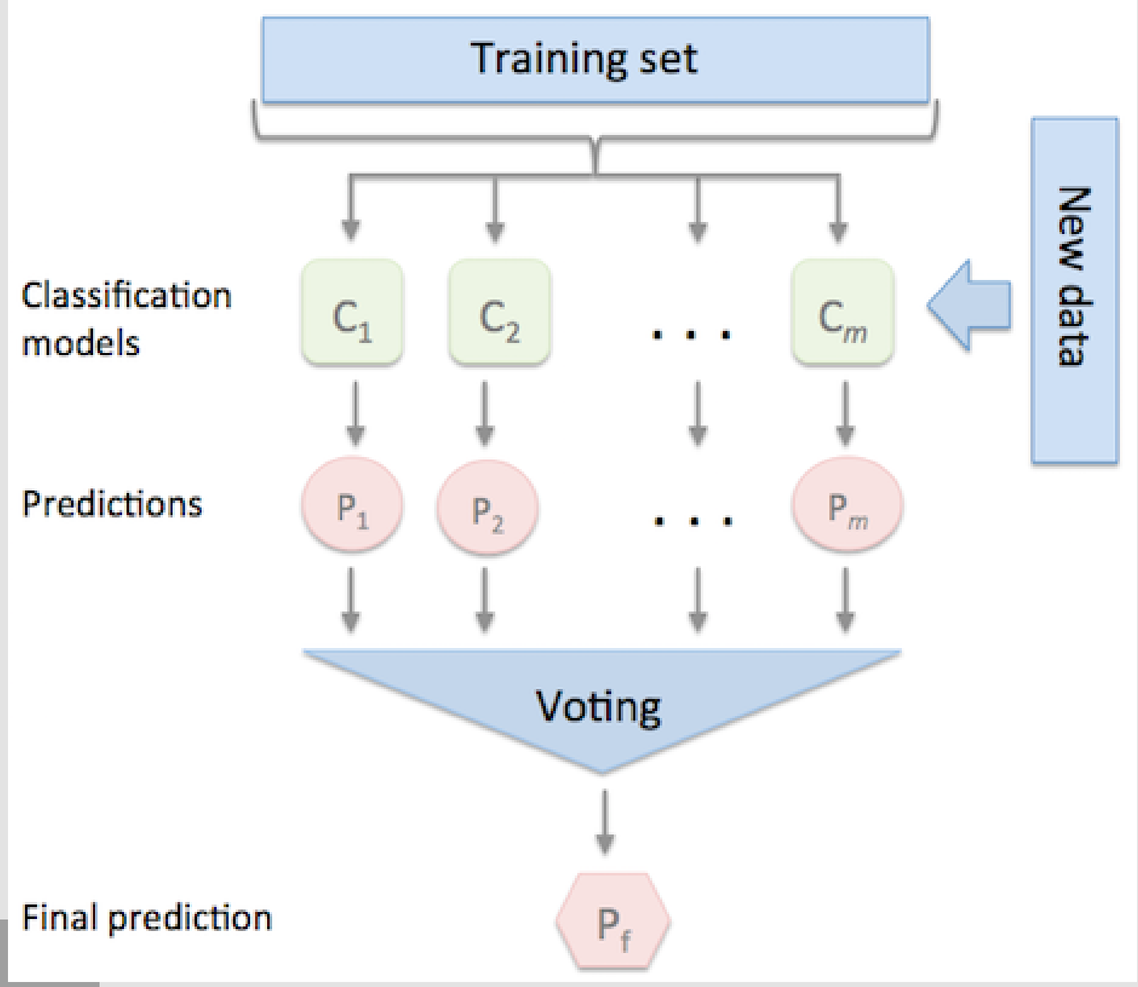


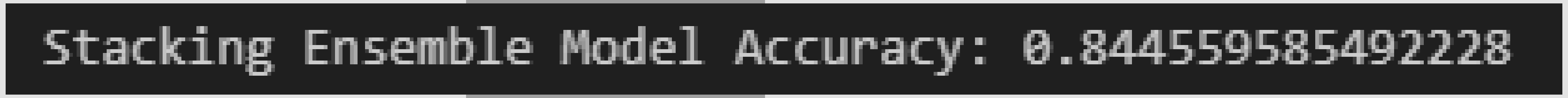
```
# Stacking Ensemble Setup
stacking_ensemble_classifier = VotingClassifier(estimators=base_classifiers, voting="hard", n_jobs=-1)

# Train Stacking Ensemble
stacking_ensemble_classifier.fit(features_train, target_train)
```

Voting Classifer

The next step is combining three models into one by using **Voting Classifier**. Every model makes a prediction and by counting, the prediction with most votes wins. This improves consistency, as if one model fails to work with data, other models will override the faulty prediction.



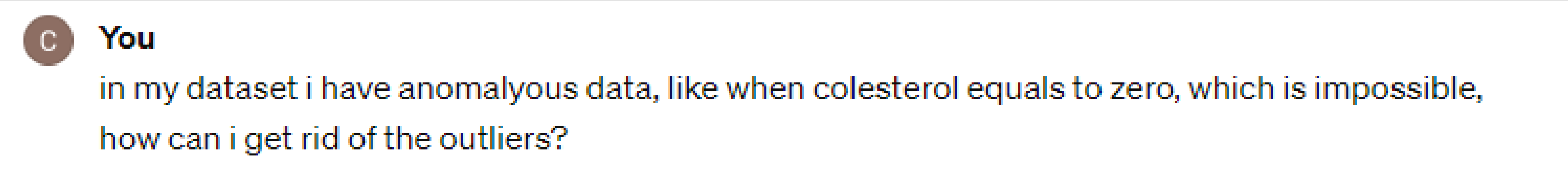


```
Stacking Ensemble Model Accuracy: 0.844559585492228
```

ChatGPT prompts

Code generated

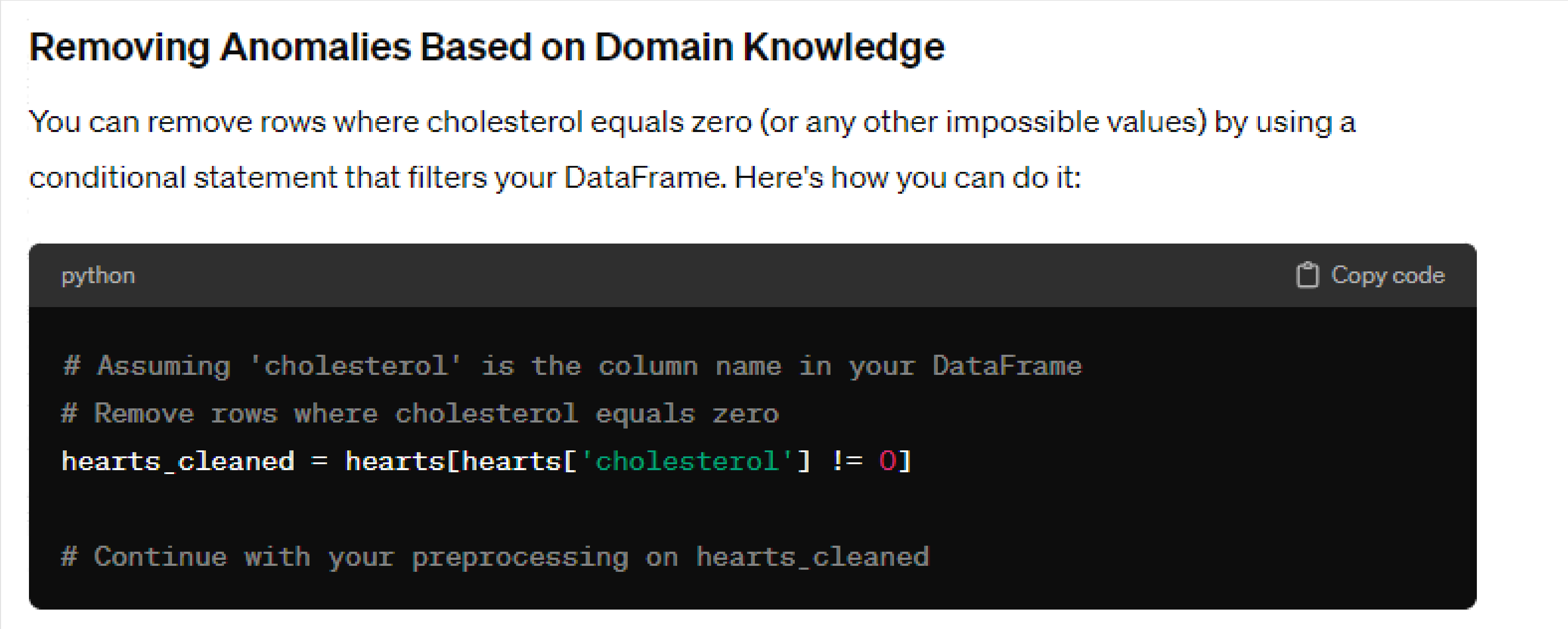
Prompt:



You

in my dataset i have anomalyous data, like when colesterol equals to zero, which is impossible, how can i get rid of the outliers?

Answer:



Removing Anomalies Based on Domain Knowledge

You can remove rows where cholesterol equals zero (or any other impossible values) by using a conditional statement that filters your DataFrame. Here's how you can do it:

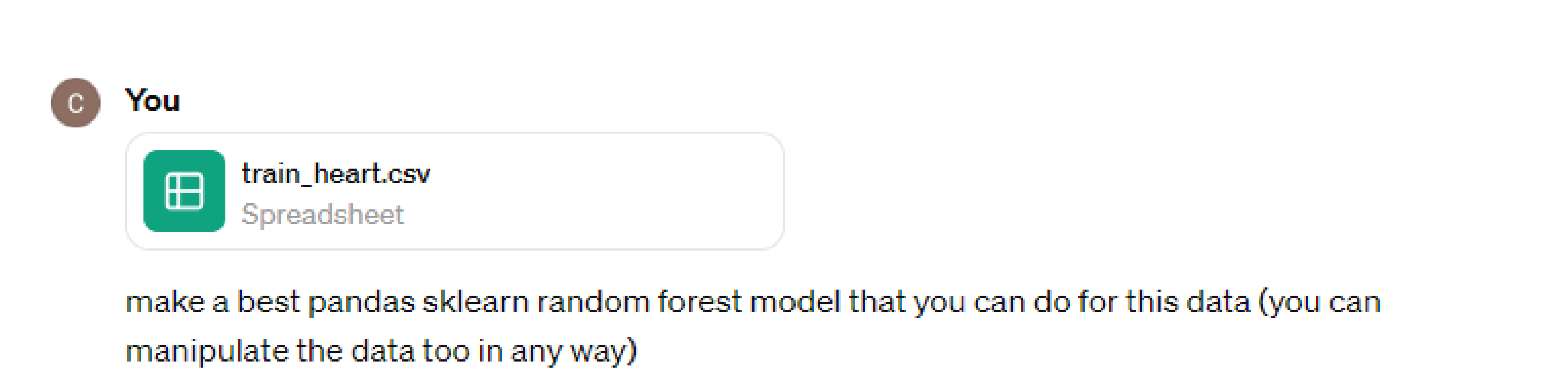
```python
# Assuming 'cholesterol' is the column name in your DataFrame
# Remove rows where cholesterol equals zero
hearts_cleaned = hearts[hearts['cholesterol'] != 0]

# Continue with your preprocessing on hearts_cleaned
```

ChatGPT prompts

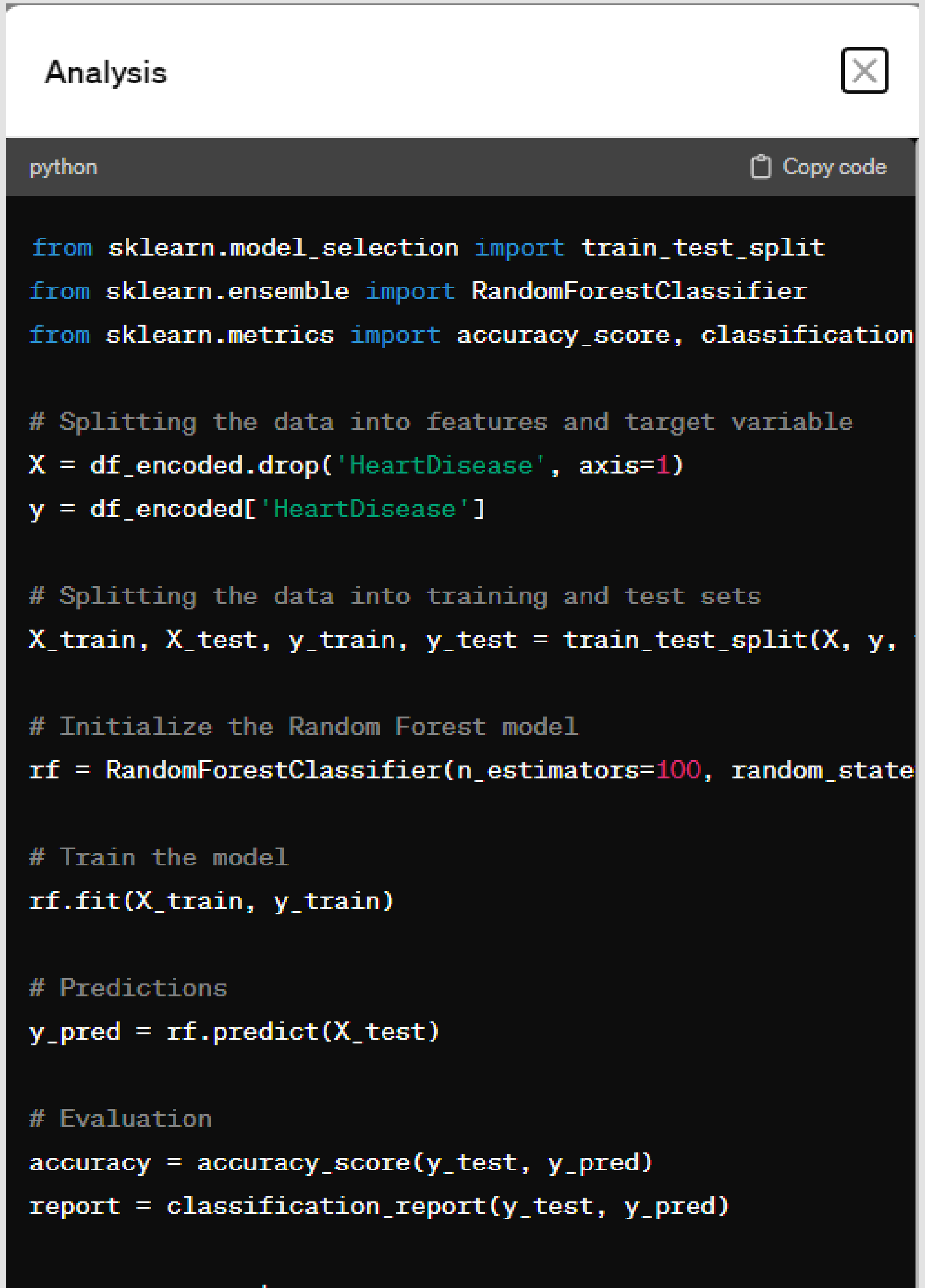
Code generated

Prompt:



make a best pandas sklearn random forest model that you can do for this data (you can manipulate the data too in any way)

Answer:



```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification

# Splitting the data into features and target variable
X = df_encoded.drop('HeartDisease', axis=1)
y = df_encoded['HeartDisease']

# Splitting the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,

# Initialize the Random Forest model
rf = RandomForestClassifier(n_estimators=100, random_state

# Train the model
rf.fit(X_train, y_train)

# Predictions
y_pred = rf.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

Same code that we already have, no improvements

THANKYOU