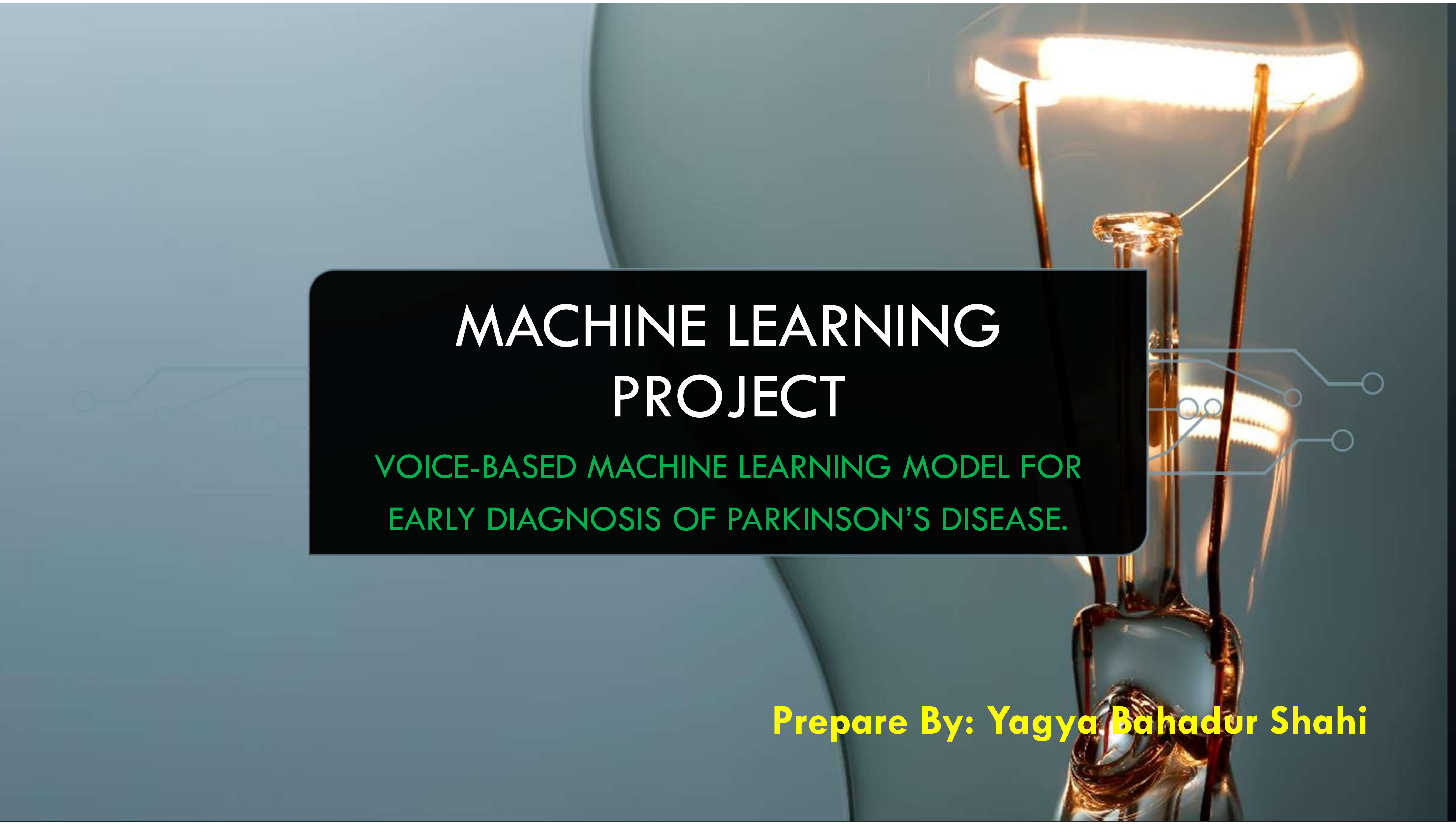
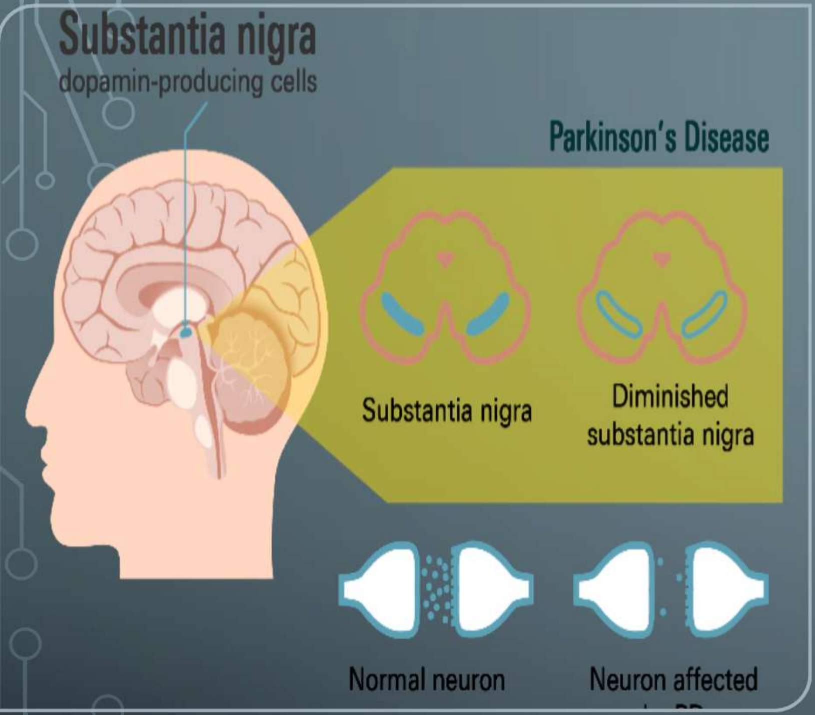


MACHINE LEARNING PROJECT

VOICE-BASED MACHINE LEARNING MODEL FOR EARLY DIAGNOSIS OF PARKINSON’S DISEASE.

Prepare By: Yagya Bahadur Shahi



- The brain is the main controller of our body. Therefore, any damage to this sensitive part will affect badly on the other organs. One of theses negative effects is Parkinson's Disease (PD). PD is movement disorder of the nervous system that worsens over time. The precise cause of PD is unknown, although some cases of PD are hereditary and can be traced to specific genetic mutations. The most common symptoms of Parkinson's disease happen because of damage to a part of the brain called the substantia nigra, which is located near the brain's base. Normally, neurons (nerve cells) in this area produce a chemical called dopamine. Dopamine helps send signals between the substantia nigra and another part of the brain called the corpus striatum. These signals are important for controlling smooth and right movements. When dopamine is lost, the brain's nerve cells start firing abnormally, which leads to difficulties with movement, causing symptoms like tremors, stiffness, and slowness.

OBJECTIVE

The primary objective of this project is to build a machine learning model that can help to detect the Parkinson's Disease (PD) at an early by analyzing people's voices.

Since PD can cause noticeable change in voice, this project uses voice data to make diagnosis easier and more affordable. By using a Support Vector Machine (SVM) model, it can detect people with Parkinson's Disease and those without it. Here is main objective for this project:

Study voice data to find pattern that are different between people with PD and those without it.

Train the SVM model to recognize these patterns.

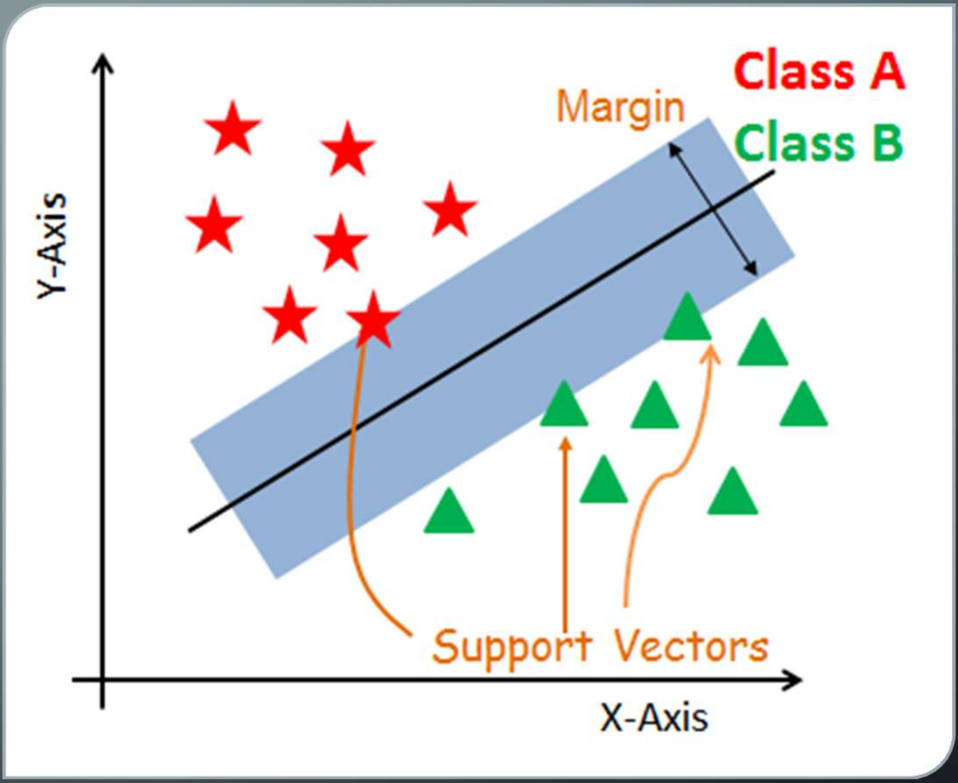
Test how accurately the model can detect PD and see if it could be useful for early diagnosis.

LIBRARY AND MODEL:

- Pandas :Pandas is a Python library for data manipulation and analysis. It provides tools for handling structured data, like reading datasets, cleaning, and organizing them into tabular formats like DataFrames for analysis.
- NumPy: NumPy is a Python library for numerical computing. It provides support for arrays, matrices, and mathematical operations, making it essential for data manipulation and computation.
- Train-Test Split: This method divides the dataset into training and testing subsets. The training set is used to train the model, while the test set evaluates its performance on unseen data.
- StandardScaler: StandardScaler standardizes data by scaling features to have a mean of 0 and a standard deviation of 1. It ensures that all features contribute equally to the model.
- SVC (Support Vector Classifier): SVC is a classifier from the SVM algorithm. It finds the optimal boundary (hyperplane) to separate classes in the dataset and is used for classification tasks.
- Accuracy Score: Accuracy Score measures the proportion of correctly classified samples in a model. It evaluates how well the model performs on the test data.

SUPPORT VECTOR MODEL(SVM):

- **Support Vector Machine (SVM) is a machine learning algorithm used to classify data, making it useful for detecting diseases like Parkinson's. It works by finding the best boundary (hyperplane) that separates data into two groups, such as healthy and Parkinson's patients. The algorithm focuses on the most important data points, called support vectors, to define this boundary. If the data is complex and not easily separable, SVM uses a technique called the kernel trick to make the data easier to classify. This helps in accurately detecting patterns and making early diagnoses.**



CODE

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

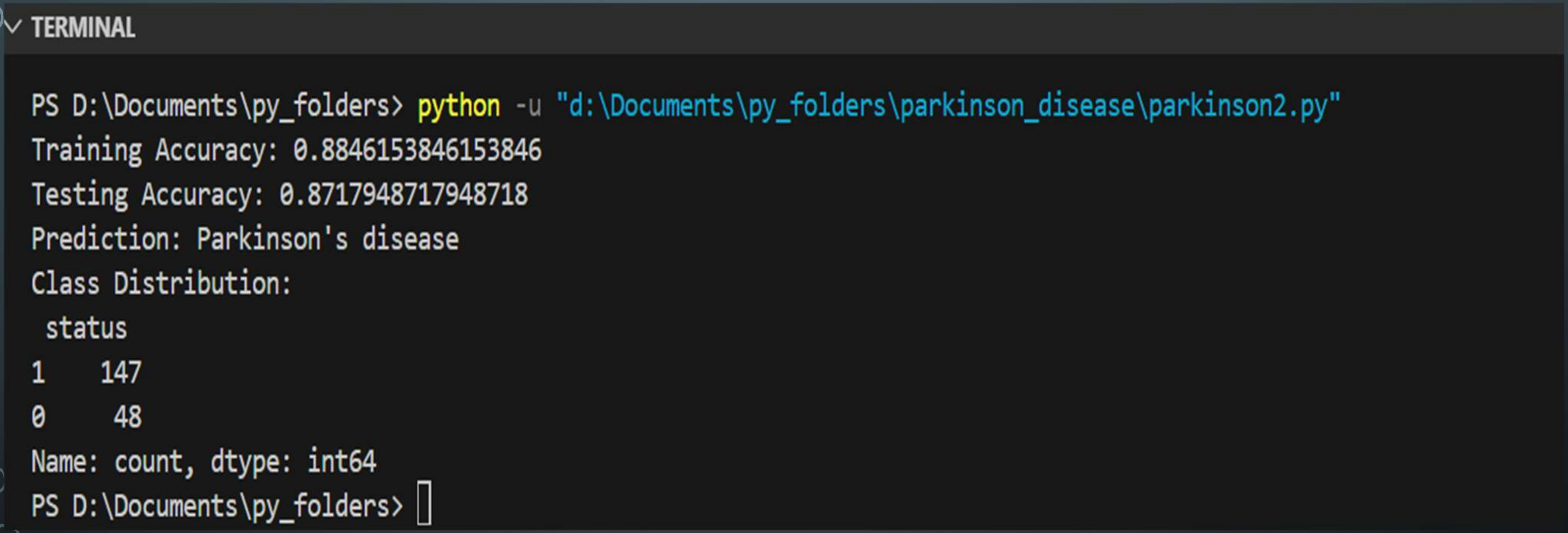
```
# Load dataset
data =
pd.read_csv("D:\\Documents\\py_folders\\parkinson_disease\\data
set.csv")

# Prepare features and target
x = data.drop(columns=['name', 'status'], axis=1)
y = data['status']

# Train-test split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=2)
```

```
# Standardize data
scaler = StandardScaler()
x_train =
scaler.fit_transform(x_train)
x_test =
scaler.transform(x_test)

# Train SVM model
model = SVC(kernel='linear',
random_state=2)
model.fit(x_train, y_train)
```

```
# Evaluate model
train_acc =
accuracy_score(y_train,
model.predict(x_train))
test_acc =

accuracy_score(y_test,
model.predict(x_test))
print("Training Accuracy:",
train_acc)
print("Testing Accuracy:",
test_acc)
```

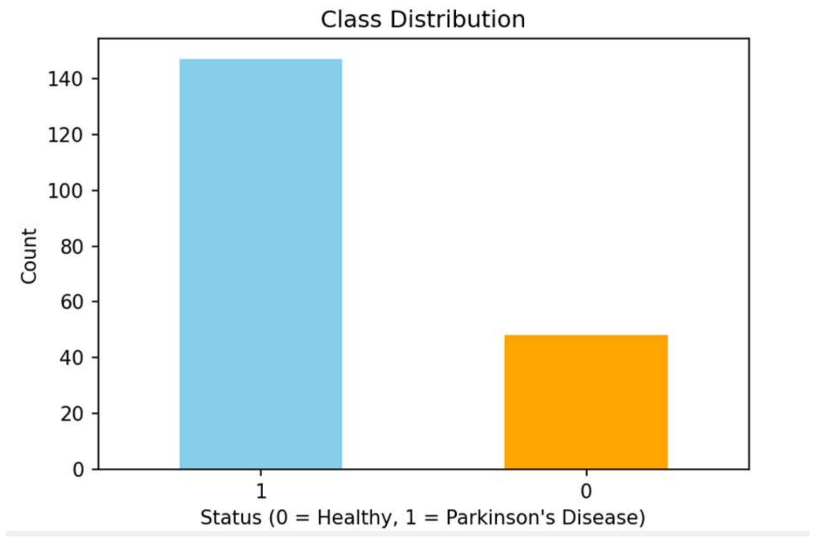
```
# Predict for new data
# Extract the first row from the dataset (excluding 'name' and 'status') as input data
input_data = data.drop(columns=['name', 'status'], axis=1).iloc[5:6]  # Keep it as a
DataFrame

# Standardize the input data using the same scaler
input_data_std = scaler.transform(input_data)

# Predict
prediction = model.predict(input_data_std)

# Output the prediction
print("Prediction for the first data row:", prediction)

# Check class distribution
class_distribution = data['status'].value_counts()
print("Class Distribution:\n", class_distribution)
```

OUTPUT:



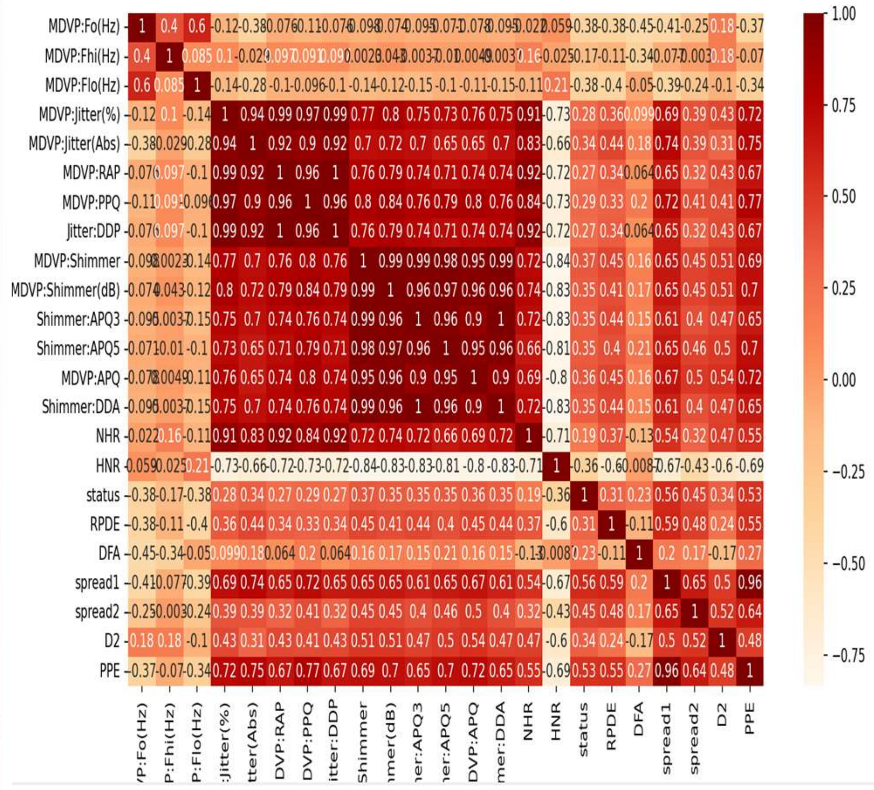
TERMINAL

```
PS D:\Documents\py_folders> python -u "d:\Documents\py_folders\parkinson_disease\parkinson2.py"
Training Accuracy: 0.8846153846153846
Testing Accuracy: 0.8717948717948718
Prediction: Parkinson's disease
Class Distribution:
 status
1    147
0     48
Name: count, dtype: int64
PS D:\Documents\py_folders>
```



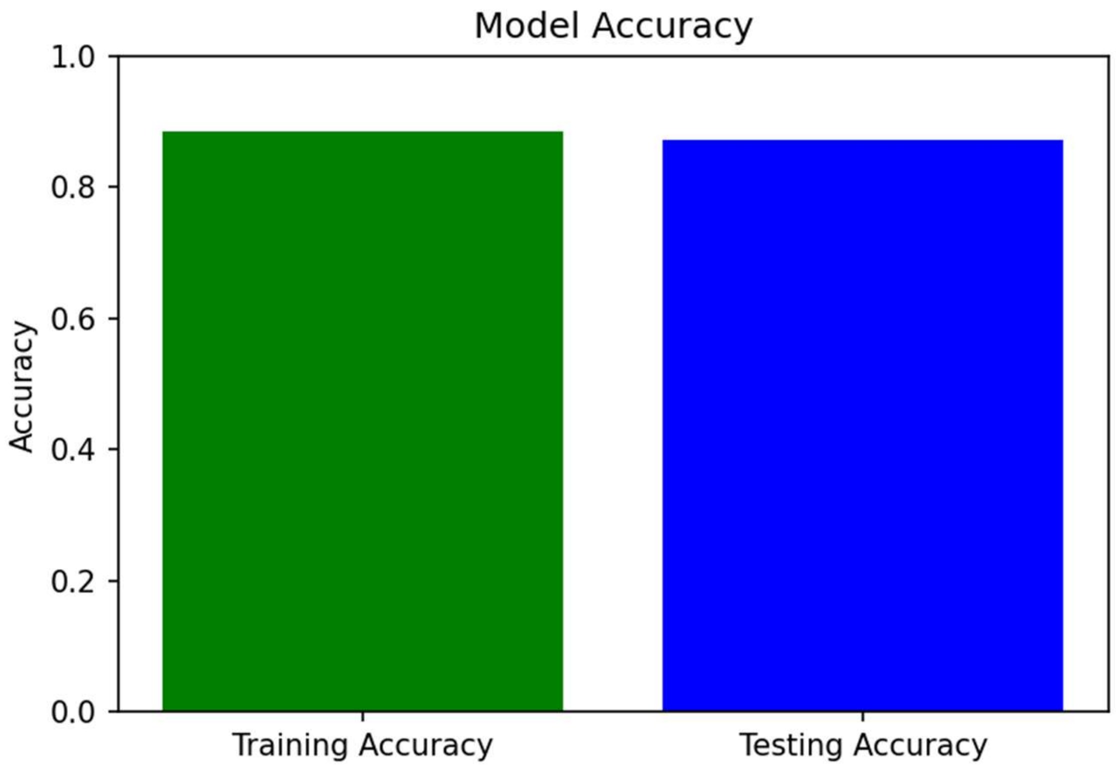
```
import matplotlib.pyplot as plt

# Visualizing class distribution

class_distribution = data['status'].value_counts()

plt.figure(figsize=(6, 4))

class_distribution.plot(kind='bar', color=['skyblue',
'orange'])

plt.title("Class Distribution")

plt.xlabel("Status (0 = Healthy, 1 = Parkinson's
Disease)")

plt.ylabel("Count")

plt.xticks(rotation=0)

plt.show()
```

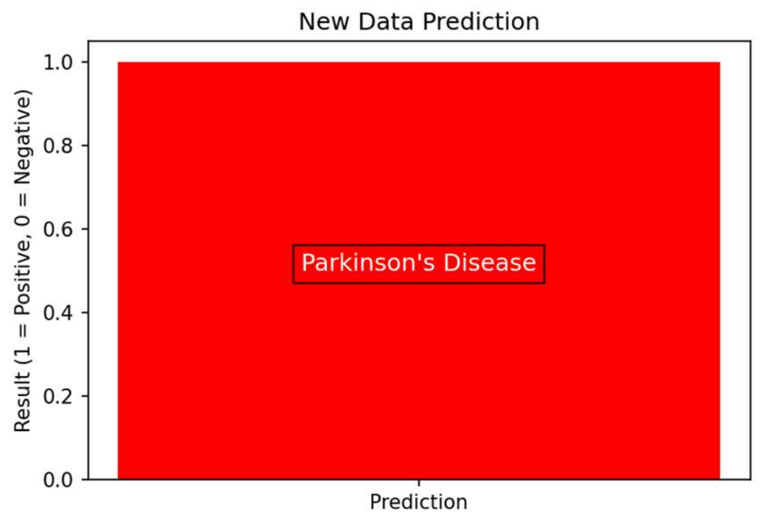
Correlation Heatmap



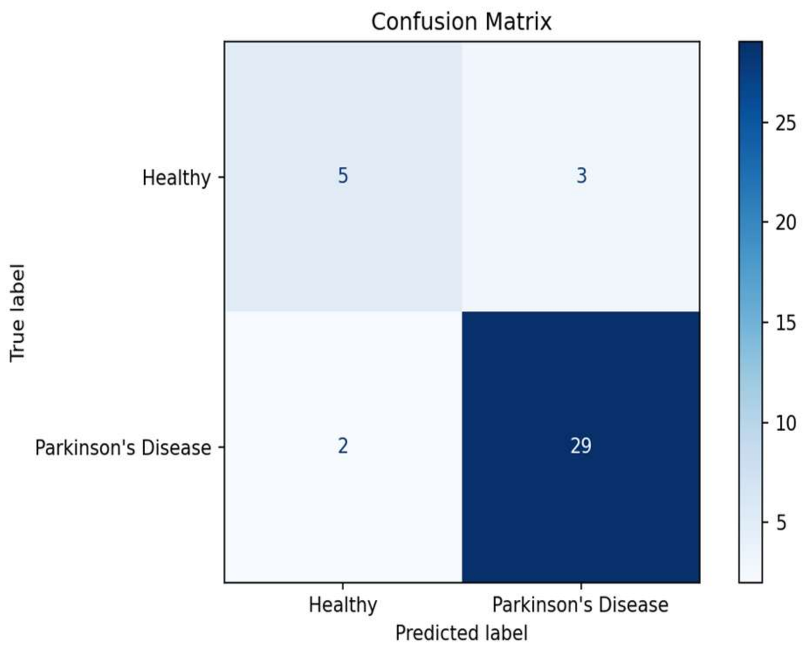
```
import seaborn as sns

# Correlation heatmap

plt.figure(figsize=(12, 10))

correlation_matrix =
data.drop(columns=['name',
'status']).corr()

sns.heatmap(correlation_matrix,
annot=False, cmap='coolwarm', cbar=True)

plt.title("Feature Correlation Heatmap")

plt.show()
```

```
# Accuracy visualization

accuracies = [train_acc,
test_acc]

labels = ['Training Accuracy',
'Testing Accuracy']


plt.figure(figsize=(6, 4))

plt.bar(labels, accuracies,
color=['green', 'blue'])

plt.ylim(0, 1)  # Accuracy is
between 0 and 1

plt.title("Model Accuracy")

plt.ylabel("Accuracy")

plt.show()
```

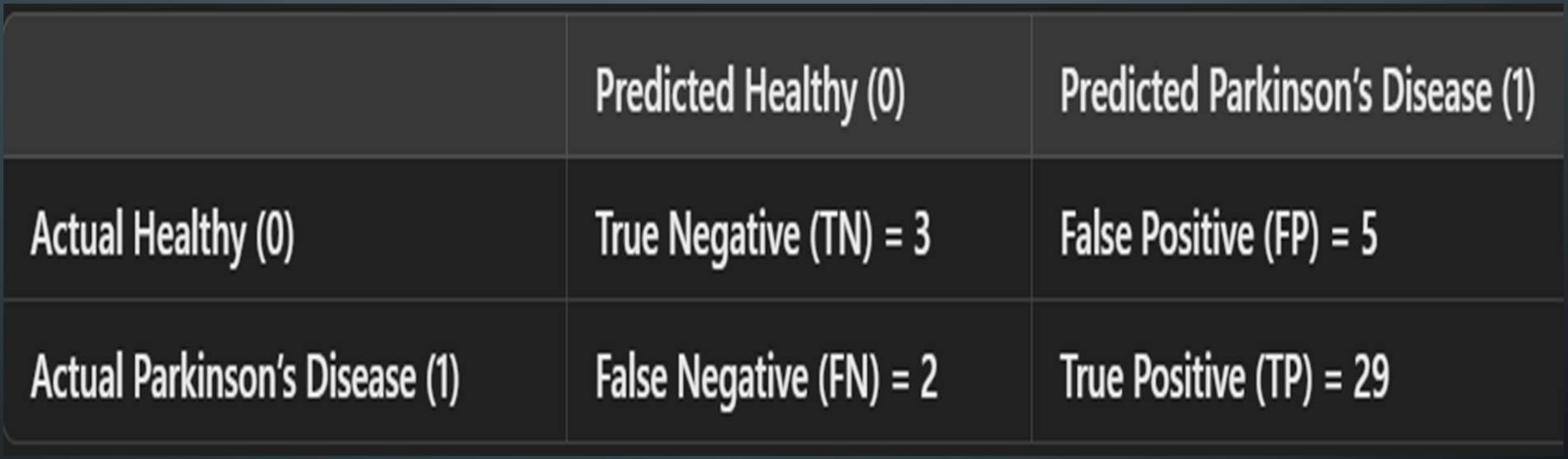




```
# Prediction visualization

result = "Parkinson's Disease" if prediction[0] == 1 else
"Healthy"

colors = ['red'] if prediction[0] == 1 else ['green']


plt.figure(figsize=(6, 4))

plt.bar(["Prediction"], [1], color=colors)

plt.title("New Data Prediction")

plt.ylabel("Result (1 = Positive, 0 = Negative)")

plt.xticks(rotation=0)

plt.text(0, 0.5, result, fontsize=12, ha='center', color='white',
bbox=dict(facecolor=colors[0], alpha=0.8))

plt.show()
```



```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Confusion matrix

cm = confusion_matrix(y_test, model.predict(x_test))

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Healthy", "Parkinson's Disease"])

disp.plot(cmap="Blues")

plt.title("Confusion Matrix")

plt.show()
```



	Predicted Healthy (0)	Predicted Parkinson's Disease (1)
Actual Healthy (0)	True Negative (TN) = 3	False Positive (FP) = 5
Actual Parkinson's Disease (1)	False Negative (FN) = 2	True Positive (TP) = 29

THANK YOU!!!!!