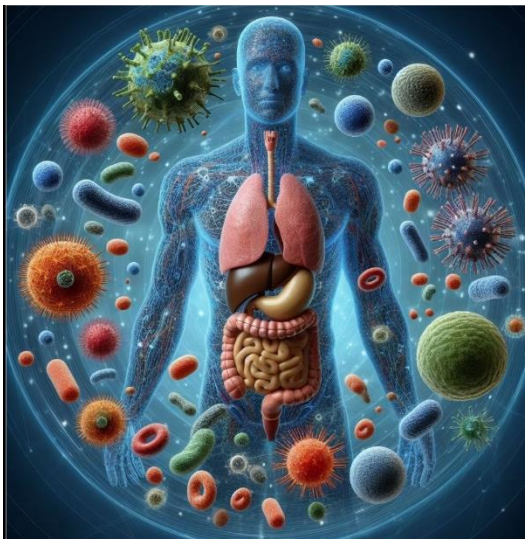
ML PROJECT REPORT



MULTIPLE DISEASE PREDICTION SYSTEM

GROUP MEMBERS	**AQSA ASIF (CS-21039)** **FATIMA SIDDIQUI (CS-21002)** **BARIRA BABAR (CS-21014)**
YEAR	**TE**
BATCH	**2021**
COURSE TITLE & **COURSE CODE**	**MACHINE LEARNING** **(CS-324)**
SUBMITTED TO	**MS. MAHNOOR MALIK**

Problem Definition:

The task is to explore a dataset, perform exploratory data analysis (EDA), and build a predictive model using machine learning algorithms.

Data Collection:

We have chosen three datasets from Kaggle for this project:

- **Diabetes Dataset**: The dataset has 303 rows and 14 columns.
- **Heart Disease Dataset:** The dataset has 195 rows and 24 columns.
- **Parkinson's Disease Dataset**

These datasets are suitable for predictive modeling and have enough features to work with.

Data Preprocessing:

We will inspect the dataset, handle any missing values, and encode categorical variables. We will also visualize the dataset to understand its structure and identify any patterns.

Exploratory Data Analysis (EDA):

The EDA process involves visualizing the distribution of numerical features and their relationship with each other

Feature Engineering:

The features are selected and scaled using StandardScaler. The target variable is also separated from the features.

Model Building:

For heart dataset:

Two models are built for this project: Logistic Regression and K-Nearest Neighbors (KNN) Classifier.Both models are trained on the training data and then used to make predictions on the test data.

Model Evaluation:

For heart dataset:

The performance of both models is evaluated on the training data to check for overfitting. The Logistic Regression model achieves an accuracy of 0.864 on the training data and 0.852 on the test data. The KNN model achieves an accuracy of 0.889 on the training data and 0.867 on the test data.

Conclusion:

For heart dataset:

Both models perform well on the dataset, with the KNN model slightly outperforming the Logistic Regression model. The models could be improved with more feature engineering or by tuning the hyperparameters. The models are then saved for future use.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.85	0.87	0.84	0.86
K-Nearest Neighbors (KNN)	0.87	0.93	0.81	0.87

Building a Predictive System (KNN Classifier)

```
input_data = (58,0,3,150,283,1,0,162,0,1,2,0,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = knn.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[1]
The Person has Heart Disease
```

Building a predictive model (logistic Regression)

```
input_data = (58,0,3,150,283,1,0,162,0,1,2,0,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = log_reg.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

For diabetes dataset :

Two models are built for this project: Logistic Regression and Decision tree Classifier. Both models are trained on the training data and then used to make predictions on the test data.

Model Evaluation:

For diabetes dataset:

The performance of both models is evaluated on the training data to check for overfitting. The Logistic Regression model achieves an accuracy of 0.77 on the training data and 0.75 on the test data. The Decision tree model achieves an accuracy of 0.79 on the training data and 0.74 on the test data.

Conclusion:

For diabetes dataset: Both models perform well on the dataset, with the Logistic model slightly outperforming the decision tree model. The models could be improved with more feature engineering or by tuning the hyperparameters. The models are then saved for future use.

Accuracy Score for LOGISTIC REGRESSION

```
# accuracy on training data
X_train_prediction_l = l.predict(X_train)
training_data_accuracy_l= accuracy_score(X_train_prediction_l, y_train)
print('Accuracy on Training data : ', round(training_data_accuracy_l,3))
```

```
Accuracy on Training data :  0.77
```

```
# Assuming l is your trained model and X_test, Y_test are your test data
X_test_prediction_l = l.predict(X_test)
test_data_accuracy_l = accuracy_score(y_test, X_test_prediction_l)
print('Accuracy on Test data:', round(test_data_accuracy_l, 3))
```

```
Accuracy on Test data: 0.753
```

Accuracy Score for decision tree

```
# Calculate accuracy for the custom Decision Tree model
def accuracy(y_true, y_pred):
    return np.sum(y_true == y_pred) / len(y_true)

train_accuracy_custom = accuracy(y_train, y_train_pred_custom)
test_accuracy_custom = accuracy(y_test, y_test_pred_custom)

print("Custom Decision Tree Training Accuracy: ", train_accuracy
print("Custom Decision Tree Test Accuracy: ", test_accuracy_cust
```

```
Custom Decision Tree Training Accuracy:  0.7915309446254072
Custom Decision Tree Test Accuracy:  0.7402597402597403
```

For Parkinson's dataset

Model Building:

For Parkinsons dataset:

Two models are built for this project: Logistic Regression and K-Nearest Neighbors (KNN) Classifier. Both models are trained on the training data and then used to make predictions on the test data.

Model Evaluation:

For Parkinsons dataset:

The performance of both models is evaluated on the training data to check for overfitting. The Logistic Regression model achieves an accuracy of 0.853 on the training data and 0.923 on the test data. The KNN model achieves an accuracy of 0.973 on the training data and 0.923 on the test data.

Conclusion:

For Parkinsons dataset:

Both models perform well on the dataset, with the KNN model slightly outperforming the Logistic Regression model. The models could be improved with more feature engineering or by tuning the hyperparameters. The models are then saved for future use.

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.855	0.914894	0.803738	0.855721
K-Nearest Neighbors	0.785	0.863636	0.710280	0.779487

Building a Predictive System (KNN Classifier)

```
input_data =(197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.0

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = knn.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Parkinsons Disease')
else:
  print('The Person has Parkinsons Disease')
```

```
[0]
The Person does not have a Parkinsons Disease
```

Building a predictive model (logistic Regression)

```
input_data = input_data = (198.234, 205.678, 193.567, 0.00345, 0.00002, 0.00189, 0.00192, 0.00567,

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = log_reg.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have Parkinsons Disease')
else:
  print('The Person has Parkinsons Disease')
```

```
[0]
The Person does not have Parkinsons Disease
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid fea
  warnings.warn(
```

Future Enhancements for Our Project

While our project has successfully implemented predictive models for the diabetes, heart disease, and Parkinson's datasets, there are several areas where we can enhance and expand our work. Here are some potential future enhancements:

Time Series Analysis

- **Objective**: Enhance predictions by incorporating time-based trends for datasets with temporal data.
- **Approach**: Implement time series models such as ARIMA, LSTM (Long Short-Term Memory networks), and Prophet.
- **Tools**: Statsmodels, TensorFlow/Keras, Facebook Prophet.

Explainable AI (XAI)

- **Objective**: Increase transparency and trust in model predictions.
- **Approach**: Utilize techniques like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) to explain model predictions.
- **Tools**: SHAP library, LIME library.