Project 5 : Diabetes Data Prediction

Source Code :

```
#Importing necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

#Importing files
from google.colab import files
#Uploading files
uploaded = files.upload()

# Load the dataset
data = pd.read_csv('diabetes.csv')

# Display the first few rows of the dataset
print(data.head())

# Basic information about the dataset
print(data.info())

# Summary statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Visualizing the distribution of outcome variable (0: No diabetes, 1: Diabetes)
sns.countplot(data['Outcome'])
plt.show()

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

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a Random Forest Classifier model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Model evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Predict for the entire dataset
predictions = model.predict(X)

# Append the predictions to the original dataset
data['Predicted_Outcome'] = predictions

# Save the data with predictions to a new CSV file
data.to_csv('diabetes_with_predictions.csv', index=False)

df1=pd.read_csv('diabetes_with_predictions.csv')
df1

#Find out correlation of "Outcome" column with other columns
correlation_matrix = df1.corr()
correlation_matrix["Outcome"]

#Find out correlation of "DiabetesPedigreeFunction" column with other columns
correlation_matrix = df1.corr()
correlation_matrix["DiabetesPedigreeFunction"]

#Find out correlation of "Insulin" column with other columns
correlation_matrix = df1.corr()
correlation_matrix["DiabetesPedigreeFunction"]

#Find out correlation of "BloodPressure" column with other columns
correlation_matrix = df1.corr()
```

```
correlation_matrix["DiabetesPedigreeFunction"]

#Find out correlation of "BMI" column with other columns
correlation_matrix = df1.corr()
correlation_matrix["BMI"]

#Find out correlation of "BMI" column with other columns
correlation_matrix = df1.corr()
correlation_matrix["Age"]

#Importing necessary libraries to calculate distance
from scipy.spatial import distance
#Find out the Minkowski distance of "Outcome" column with "Pregnancies"
column
mink_dist1 =
distance.minkowski(df1["Outcome"].astype(float),df1["Pregnancies"].astype(float
),p=1)
print(mink_dist1)
#Find out the Minkowski distance of "Outcome" column with "Glucose" column
mink_dist2 =
distance.minkowski(df1["Outcome"].astype(float),df1["Glucose"].astype(float),p=
1)
print(mink_dist2)
#Find out the Minkowski distance of "Outcome" column with "BloodPressure"
column
mink_dist3 =
distance.minkowski(df1["Outcome"].astype(float),df1["BloodPressure"].astype(flo
at),p=1)
print(mink_dist3)
#Find out the Minkowski distance of "Outcome" column with "SkinThickness"
column
mink_dist4 =
distance.minkowski(df1["Outcome"].astype(float),df1["SkinThickness"].astype(flo
at),p=1)
print(mink_dist4)
#Find out the Minkowski distance of "Outcome" column with "Insulin" column
mink_dist5 =
distance.minkowski(df1["Outcome"].astype(float),df1["Insulin"].astype(float),p=1)
print(mink_dist5)
#Find out the Minkowski distance of "Outcome" column with "BMI" column
```

```
mink_dist6 =
distance.minkowski(df1["Outcome"].astype(float),df1["BMI"].astype(float),p=1)
print(mink_dist6)
#Find out the Minkowski distance of "Outcome" column with
"DiabetesPedigreeFunction" column
mink_dist7 =
distance.minkowski(df1["Outcome"].astype(float),df1["DiabetesPedigreeFunction
"].astype(float),p=1)
print(mink_dist7)
#Find out the Minkowski distance of "Outcome" column with "Age" column
mink_dist8 =
distance.minkowski(df1["Outcome"].astype(float),df1["Age"].astype(float),p=1)
print(mink_dist8)
```

Program Explanation :

Let's go through the program step by step :

1. **Importing Necessary Libraries :**
   - **pandas** : Data manipulation library
   - **numpy** : Numerical operations library
   - **seaborn**, **matplotlib.pyplot** : Data visualization libraries
   - **train_test_split** : Function to split data into training and testing sets from **sklearn.model_selection**
   - **RandomForestClassifier**: Random Forest classifier from **sklearn.ensemble**
   - **accuracy_score**, **confusion_matrix**, **classification_report**: Evaluation metrics from **sklearn.metrics**
   - **files** from **google.colab**: For file upload in Google Colab
2. **Uploading Files :**
   - Utilizing the **files.upload()** function to upload files in a Google Colab environment.
3. **Loading the dataset :**
   - Reading a CSV file named 'diabetes.csv' into a Pandas DataFrame named **data**.
4. **Exploratory Data Analysis (EDA) :**
   - Displaying the first few rows of the dataset using **data.head()**.
   - Providing basic information about the dataset using **data.info()**.
   - Displaying summary statistics using **data.describe()**.
   - Checking for missing values using **data.isnull().sum()**.
   - Visualizing the distribution of the outcome variable using **sns.countplot()**.
5. **Data Preparation :**
   - Splitting the data into features (**X**) and the target variable (**y**).

- Splitting the data into training and testing sets using **train_test_split()**.

6. **Model Building :**
   - Creating a Random Forest Classifier model with 100 estimators and training it using the training set.
7. **Model Evaluation :**
   - Predicting on the test set and evaluating the model using accuracy, confusion matrix, and classification report.
8. **Predictions and Saving :**
   - Predicting the outcomes for the entire dataset and appending the predictions as a new column.
   - Saving the dataset with predictions to a new CSV file named 'diabetes_with_predictions.csv'.
9. **Reading the Saved Data :**
   - Reading the saved CSV file into a new DataFrame (**df1**).
10. **Correlation Analysis :**
    - Finding the correlation of the "Outcome" column with other columns in the dataset.
11. **Minkowski Disance Calculation :**
    - Importing the necessary library (**scipy.spatial.distance**) for calculating distances.
    - Calculating Minkowski distance of "Outcome" column with each of the other columns in the dataset.
12. **Printing Minkowski Distances :**
    - Printing the Minkowski distances for each column.

This program is a comprehensive data analysis and machine learning pipeline for a diabetes dataset. It covers data loading, exploration, model training, evaluation, prediction, and distance calculation. The Minkowski distances are calculated for each column's correlation with the "Outcome" column, providing insights into the relationships between variables.