AI-BASED DIABETES PREDICTION SYSTEM

TEAM MEMBER
510521106015: RUBA SHREE.J

PHASE 5 SUBMISSION DOCUMENT

PROJECT TITLE: DIABETES PREDICTION SYSTEM

PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION

TOPIC: In this section we will document the complete project and prepare it for submission

DIABETES PREDICTION SYSTEM

INTRODUCTION:

An AI-Based Diabetes Prediction System represents a groundbreaking approach in healthcare that leverages the power of artificial intelligence to enhance early detection and management of diabetes.

By analyzing an individual's medical data,lifestyle and genetic factors,this innovative system can provide personalized risk assessments and recommendations.

This not only empowers individuals to make informed decisions about their health but also offers healthcare professionals valuable insights for more effective diabetes prevention and care.

In this intoduction,we''ll delve deeper into the development,significance,and potential of AI in predicting and managing diabetes.

DATASET LINK: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Here is a list of tools and software commonly used in the process:

1.**Programming Language:**

Python is the most popular language for machine learning due to its extensive libraries and frameworks.You can use libraries like NumPy,Pandas,Scikit-learn and more.

2.Integrated Development Environment(IDE):

Choose an IDE for coding and running machine learning experiments.Some popular options include Jupyter Notebook,Google colab,or traditional IDEs like PyCharm.

3.Machine Learning Libraries:

You'll need various machine learning libraries,including:

Scikilt-learn for building and evaluating machine learning models.

Tensor flow or PyTorch for deep learning,if needed

XGBoost,LightGBM or CatBoost for gradient boosting models

4. Data Visualizing tools:

Tools like Matplotlib,Seaborn,or plotly are essential for data exploration and visualization.

5.Data Preprocessing Tools:

Libraries like pandas help with data cleaning,manipulation and preprocessing.

6.Data Collection and Storage:

Depending upon your data source,you might need web scraping tools(eg.Beautifulsoup or scopy)or databases (eg.SQLite, PostgreSQL)for data storage.

7.Version Control:

Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

8.Notebooks and Documentation:

Tools for documenting your work,such as Jupyter Noteboks pr Markdown for creating README files and documentation.

9.Hyperparameter Tuning:

Tools like GridSearchCV or randomizedSearchCV from scikit-learn can help with hyperparameter tuning.

10.Web Development Tools(for Deployment):

If you can plan to create a web application for model deployment,knowledge of web development tools like Flask or Django for backend development,and HTML,CSS and Javascript for the front end cann be useful.

11.Cloud Services(for Scalability):

For large scale applications,cloud platforms like AWS,Google Cloud,or Azure can provide scalable computing and storage resources.

DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT:

Design thinking can be a valuable approach when developing an AI-based diabetes prediction system. Here's how it can be applied:

1. **Empathize**:Understand the needs and concerns of potential users, such as individuals at risk of diabetes, healthcare professionals, and caregivers.Conduct interviews, surveys, and observations to gain insights into their experiences and pain points related to diabetes prediction and management.

2.**Define**:Clearly define the problem you aim to solve with the AI system, taking into account the insights gained during the empathize stage.Identify the key challenges in diabetes prediction, early intervention, and personalized care.

3.Ideate:Brainstorm potential AI solutions that can address the defined problem. Consider factors like data sources, algorithms, user interfaces, and feedback mechanisms.Encourage creative thinking to generate a variety of ideas.

4.Prototype:Develop low-fidelity prototypes of the AI system, including sample interfaces and basic algorithms.These prototypes can help visualize the user experience and test the feasibility of the solution.

5. Test:Gather feedback from potential users, healthcare professionals, and stakeholders by allowing them to interact with the prototype.Evaluate the effectiveness of the AI-based diabetes prediction system in addressing the defined problem.Make iterations based on the feedback received, refining the system as necessary.

- Throughout this design thinking process, collaboration between data scientists, healthcare experts, and user experience designers is crucial to create a user-friendly and effective AI system for diabetes prediction. The iterative nature of design thinking allows for continuous improvement and ensures that the final system aligns with the real needs and concerns of its users.

DESIGN INTO INNOVATION:

Data collection and preprocessing are critical steps in developing an AI-based diabetes prediction system. Here's a high-level overview of the process:

Data Collection:

1.Identify Data Sources: Gather data from various sources, including electronic health records (EHRs), wearable devices, mobile apps, and patient surveys.Collaborate with healthcare institutions, research organizations, and diabetes clinics to access relevant datasets.

2.Select Relevant Features: Choose data attributes that are essential for diabetes prediction, such as glucose levels, family medical history, lifestyle factors, and genetic information.

3.Ensure Data Quality:Address issues like missing values, outliers, and data inconsistencies.Validate data integrity to ensure accuracy and reliability.

4.Data Privacy and Security:Comply with data privacy regulations (e.g., HIPAA) when handling sensitive healthcare data.Implement robust security measures to protect patient information.

DATA PREPROCESSING:

1.Data Cleaning: Handle missing data through techniques like imputation or removal of incomplete records.Address outliers that might skew the model by evaluating and potentially transforming extreme values.

2.Feature Engineering: Create new features or transform existing ones to extract more meaningful information. For instance, calculate the average glucose level over a certain period.

3.Normalization and Standardization: Normalize or standardize numerical features to bring them to a common scale. This ensures that features with different units or scales do not dominate the model.

4.One-Hot Encoding: Convert categorical variables into binary (0/1) vectors through one-hot encoding. This allows the model to work with categorical data.

5.Data Splitting:Split the dataset into training, validation, and test sets. This enables model training, validation, and evaluation on distinct datasets.

6.Handling Class Imbalance: If the dataset has an imbalance in the distribution of diabetes and non-diabetes cases, apply techniques like oversampling, undersampling, or synthetic data generation to address this issue.

7.Time Series Data Handling:If dealing with time-series data (e.g., continuous glucose monitoring), consider techniques such as lag features and rolling statistics to capture temporal patterns.

8.Dimensionality Reduction: Apply dimensionality reduction methods like Principal Component Analysis (PCA) if dealing with high-dimensional data.

9.Data Augmentation (Optional):Generate additional synthetic data points to enhance the training dataset, especially if the available data is limited.

Data collection and preprocessing are vital for building accurate and reliable AI models for diabetes prediction. The quality and suitability of the data directly impact the performance and effectiveness of the predictive system.

Feature Engineering:

Select relevant features that can influence diabetes prediction.Transform categorical data into numerical format using techniques like one-hot encoding.Normalize or scale numerical features to ensure they have the same impact on the model.

Model Selection:

_ Choose an appropriate machine learning or deep learning model for prediction. Common choices include logistic regression, decision trees, random forests, support vector machines, or neural networks.

Model Training:

Split your dataset into training and testing sets to evaluate the model's performance.Train your chosen model on the training data.

Hyperparameter Tuning:

Optimize the model's hyperparameters to improve its predictive accuracy.

Evaluation:

Use metrics such as accuracy, precision, recall, F1-score, and ROC-AUC to assess the model's performance.Implement cross-validation for a robust evaluation.

Deployment:

Develop a user-friendly interface for the diabetes prediction system.Deploy the model on a cloud platform or a server.

Continuous Monitoring and Updates:

Regularly update the model with new data to improve accuracy and adapt to changing trends.Implement monitoring for model drift and retraining triggers.

Privacy and Ethics:

Ensure that patient data is anonymized and privacy-compliant.Adhere to ethical guidelines and regulations related to healthcare AI.

User Education:

Provide users with information on how to interpret the system's predictions and encourage them to consult healthcare professionals for diagnosis and treatment.

Data Preprocessing:

Program:

Input:

```
#Transform the data to integer
Data[“Diabetes_binary”] = data[“Diabetes_binary”].astype(int)
data[“HighBP”] = data[“HighBP”].astype(int)
data[“HighChol”] = data[“HighChol”].astype(int)
data[“CholCheck”] = data[“CholCheck”].astype(int)
data[“BMI”] = data[“BMI”].astype(int)
data[“Smoker”] = data[“Smoker”].astype(int)
data[“Stroke”] = data[“Stroke”].astype(int)
data[“HeartDiseaseorAttack”] =
data[“HeartDiseaseorAttack”].astype(int)
```

```
Data[“Veggies”] = data[“Veggies”].astype(int)
data[“HvyAlcoholConsump”] = data[“HvyAlcoholConsump”].astype(int)
data[“AnyHealthcare”] = data[“AnyHealthcare”].astype(int)
data[“NoDocbcCost”] = data[“NoDocbcCost”].astype(int)
data[“GenHlth”] = data[“GenHlth”].astype(int)
data[“MentHlth”] = data[“MentHlth”].astype(int)
data[“PhysHlth”] = data[“PhysHlth”].astype(int)
data[“DiffWalk”] = data[“DiffWalk”].astype(int)
data[“Sex”] = data[“Sex”].astype(int)
data[“Age”] = data[“Age”].astype(int)
data[“Education”] = data[“Education”].astype(int)
```

Output:

```
Data.info()
 <class ‘pandas.core.frame.DataFrame’>
 RangeIndex: 253680 entries, 0 to 253679
Data columns (total 22 columns):
# Column Non-Null Count Dtype
 --- ------ -------------- -----
0 Diabetes_binary 253680 non-null int64
 1 HighBP 253680 non-null int64
 2 HighChol 253680 non-null int64
 3 CholCheck 253680 non-null int64
4 BMI 253680 non-null int64
 5 Smoker 253680 non-null int64
```

6 Stroke 253680 non-null int64
7 HeartDiseaseorAttack 253680 non-null int64
8 PhysActivity 253680 non-null int64
9 Fruits 253680 non-null int64
10 Veggies 253680 non-null int64
11 HvyAlcoholConsump 253680 non-null int64
12 AnyHealthcare 253680 non-null int64
13 NoDocbcCost 253680 non-null int64
14 GenHlth 253680 non-null int64
15 MentHlth 253680 non-null int64
16 PhysHlth 253680 non-null int64
17 DiffWalk 253680 non-null int64

Validation and Testing:

Validate the system's predictions with real-world patients and gather feedback for improvements.

Documentation and Reporting:

Maintain detailed documentation of the project, including data sources, model architecture, and deployment procedures.Prepare reports on the system's performance and findings.

Future Enhancements:

Consider adding features like risk stratification, personalized recommendations, or integration with wearable devices for real-time monitoring.

Collaboration:

Collaborate with healthcare professionals, data scientists, and domain experts to enhance the system's accuracy and usability.

BUILD LOADING AND PREPROCESSING THE DATASET:

Creating a dataset for an AI-based diabetes prediction system typically involves two main steps: LOADING and preprocessing. Below are the general steps for these processes:

1.LOADING the Dataset:

The first step is to obtain a dataset that contains relevant information about diabetes and related factors. You can find such datasets from various sources like government health agencies, research organizations, or open data repositories. For this example, let's assume you have a CSV file named "diabetes_data.csv."

```
Import pandas as pd
# Load the dataset from a CSV file
diabetes_data = pd.read_csv("diabetes_data.csv")
# Display the first few rows to inspect the data
print(diabetes_data.head())
```

2. Preprocessing the Dataset:

Data preprocessing is crucial to prepare the dataset for machine learning. Here are some common preprocessing steps:

2.1. Handling Missing Values:

```
# Check for missing values
print(diabetes_data.isnull().sum())
# Handle missing values, for example, by imputing with mean or median
diabetes_data['feature_name'].fillna(diabetes_data['feature_name'].mean(), inplace=True)
```

2.2. Encoding Categorical Variables:

If your dataset contains categorical variables (e.g., 'Gender' or 'Smoking'), you need to encode them numerically.

```
# Use one-hot encoding for categorical variables
diabetes_data = pd.get_dummies(diabetes_data, columns=['Categorical_feature'])
```

2.3. Scaling Numerical Features:

Scaling features helps algorithms converge faster and prevents some features from dominating others.from sklearn.preprocessing.

```
import StandardScaler
scaler = StandardScaler()
diabetes_data[['Age', 'BMI']] = scaler.fit_transform(diabetes_data[['Age', 'BMI']])
```

2.4. Splitting the Data:

Divide the data into training and testing sets for model evaluation.from sklearn.

```
model_selection import train_test_split
X = diabetes_data.drop('Target_variable', axis=1)
y = diabetes_data['Target_variable']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2.5. Balancing the Dataset (if needed):

If your dataset is imbalanced, you might need to balance it to avoid model bias.

2.6. Saving the Preprocessed Data:

Once the preprocessing is complete, you can save the preprocessed data for later use.

diabetes_data.to_csv(“preprocessed_diabetes_data.csv”, index=False)

☆These are the fundamental steps for LOADING and preprocessing a dataset for an AI-based diabetes prediction system. Depending on the specifics of your data, you may need to perform additional data cleaning and feature engineering steps.

PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING AND EVALUATION:

Feature Engineering:

As mentioned earlier, feature engineering is crucial. It involves creating new features or transforming existing ones to provide meaningful information for your model.

EDA:

Input:

```
#using heatmap to understand correlation better in dataset data
#Heatmap of correlation
plt.figure(figsize = (20,10))
sns.heatmap(data.corr(),annot=True , cmap ='YlOrRd' )
plt.title("correlation of feature")
```

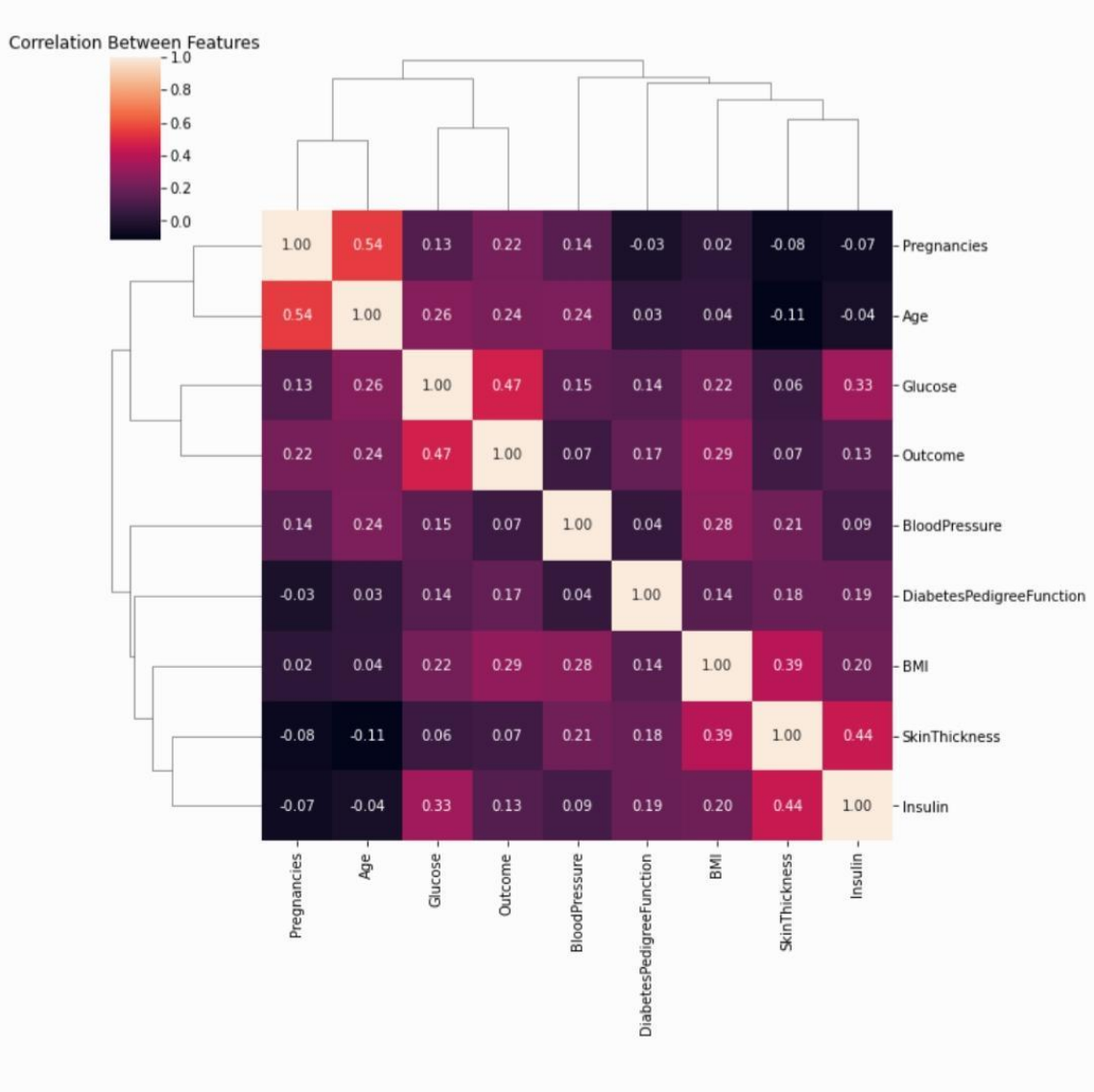
Output:

text(0.5, 1.0, ‘correlation of feature’)

Input:

```
# I am visualizing the correlation of the dataset with the seaborn library.
Corr_matrix = data.corr()
sns.clustermap(corr_matrix, annot = True, fmt = “.2f”)
plt.title(“Correlation Between Features”)
plt.show()
```

Output:

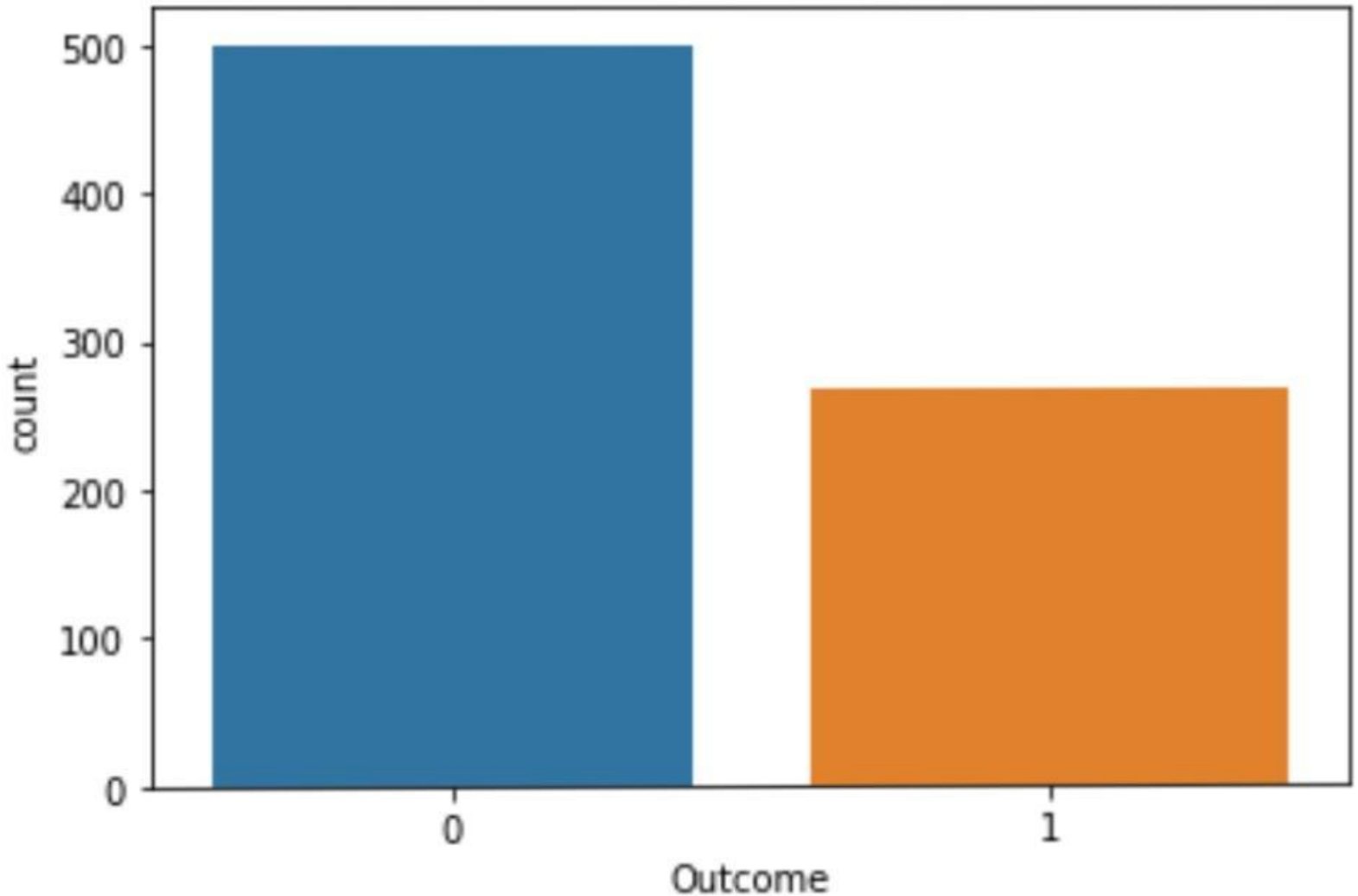

Correlation Between Features
1.0
0.8
0.6
0.4
0.2
0.0
1.00 0.54 0.13 0.22 0.14 -0.03 0.02 -0.08 -0.07 Pregnancies
0.54 1.00 0.26 0.24 0.24 0.03 0.04 -0.11 -0.04 Age
0.13 0.26 1.00 0.47 0.15 0.14 0.22 0.06 0.33 Glucose
0.22 0.24 0.47 1.00 0.07 0.17 0.29 0.07 0.13 Outcome
0.14 0.24 0.15 0.07 1.00 0.04 0.28 0.21 0.09 BloodPressure
-0.03 0.03 0.14 0.17 0.04 1.00 0.14 0.18 0.19 DiabetesPedigreeFunction
0.02 0.04 0.22 0.29 0.28 0.14 1.00 0.39 0.20 BMI
-0.08 -0.11 0.06 0.07 0.21 0.18 0.39 1.00 0.44 SkinThickness
-0.07 -0.04 0.33 0.13 0.09 0.19 0.20 0.44 1.00 Insulin
Pregnancies
Age
Glucose
Outcome
BloodPressure
DiabetesPedigreeFunction
BMI
SkinThickness
Insulin

Input:

```
# I am visualizing the correlation of the dataset with the seaborn library.

# I check Outcome data in dataset.

Plt.figure()

sns.countplot(data.Outcome)

print(data.Outcome.value_counts())
```

Output:

```
0    500

1    268

Name: Outcome, dtype: int64
```

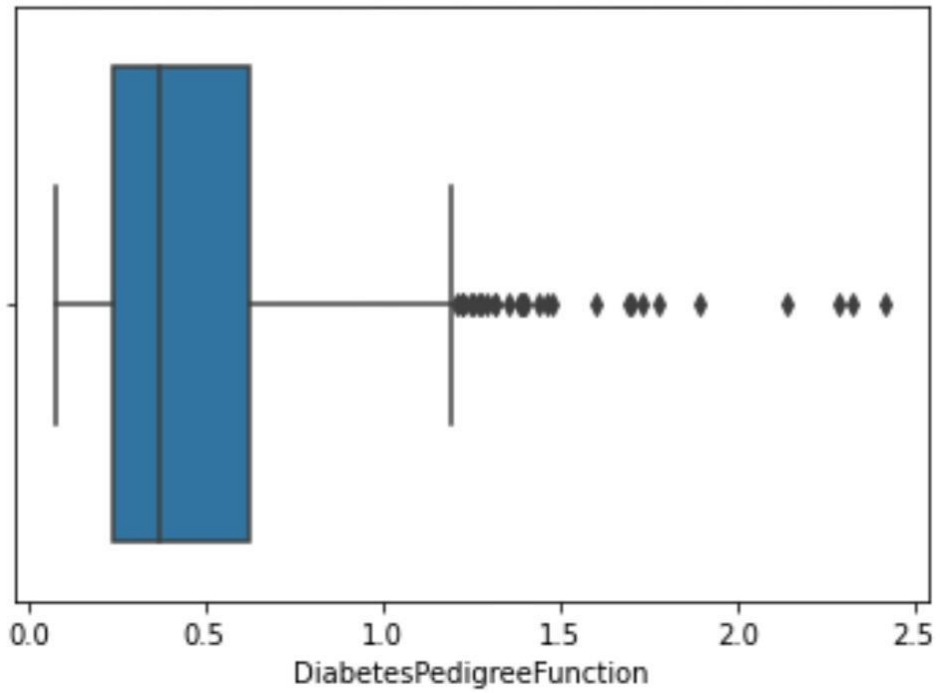

500
400
300
200
100
0
count
0
1
Outcome

Outer Values:

INPUT:

```
For c in data.columns:
    plt.figure()
    sns.boxplot(x  = c, data = data, orient = “v”)
```

OUTPUT:

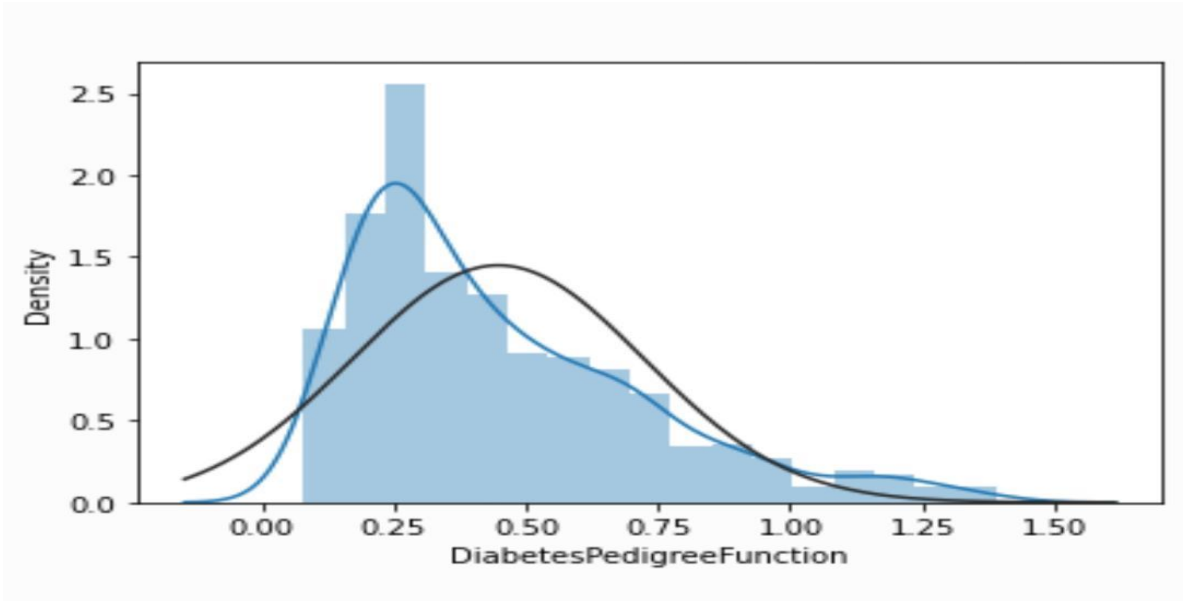


FEATURE ENGINEERING:

INPUT:

```
For i in data.columns:
 plt.figure()
    sns.distplot(data[i], fit = norm)
    plt.show()
```

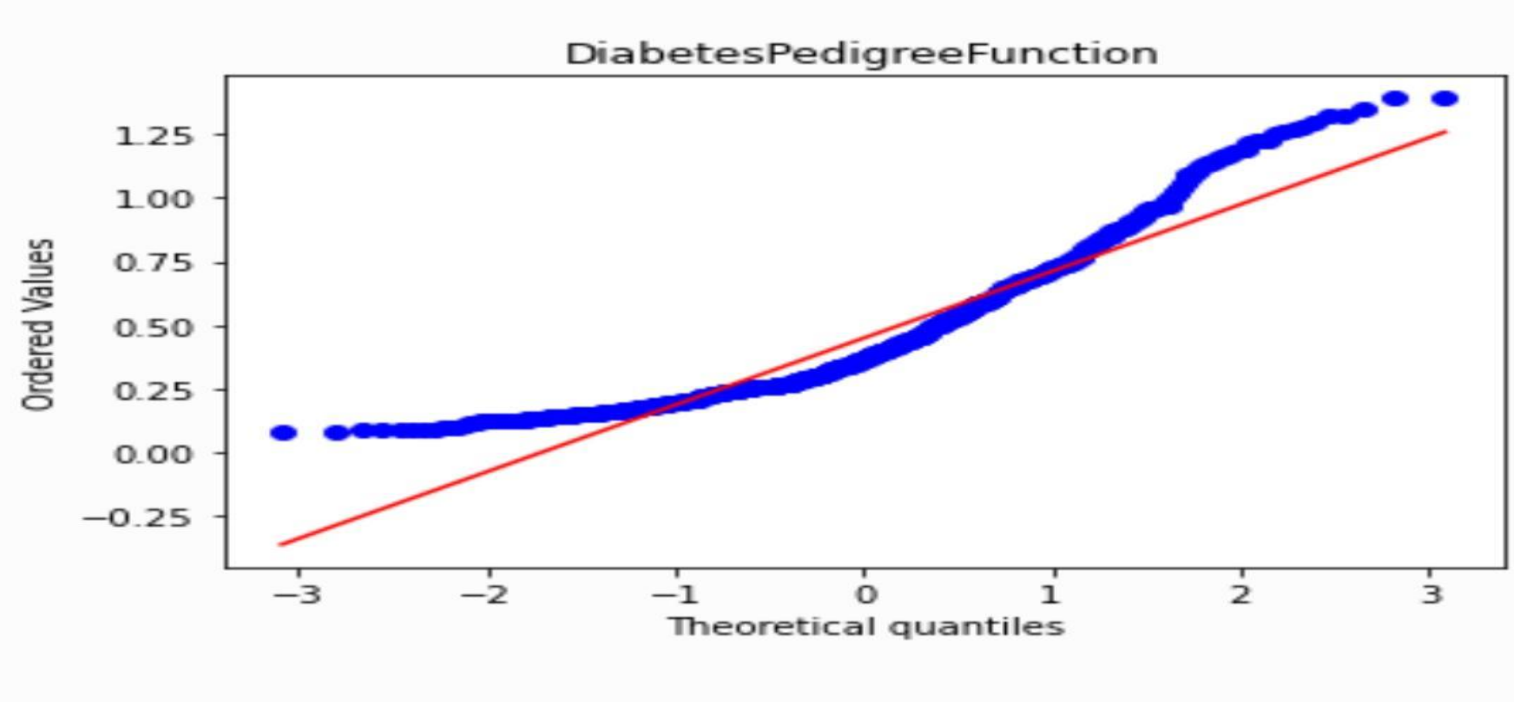
OUTPUT:



INPUT:

```
For i in data.columns:
    plt.figure()
    stats.probplot(data[i], plot = plt)
    plt.title(i)
    plt.show()
```

OUTPUT:



MACHINE LEARNING CLASSIFICATION MODELS:

INPUT:

```
# Import Naive Bayes Classification models
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
# Import Support Vector Machine Classification models
from sklearn.svm import SVC
# Import Logistic Regression Models
from sklearn.linear_model import LogisticRegression
# Import K-Nearest Neighbors(KNN) Models
from sklearn.neighbors import KNeighborsClassifier
# Import Decision Tree Classification Models
from sklearn.tree import DecisionTreeClassifier
# Import Random Forest Classification Models
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

Input:

```
# I create a dataframe for the ML models.

ResoltML_Data = pd.DataFrame(columns = [“Model_Name”, “SS_Score”, “RS_Score”])
```

INPUT:

```
Model_Name = [“GNB”,”BNB”,”SVC”,”LR”,”KNN”,”DTC”,”RFC”] # Abridgment Names of Models

ResoltML_Data[“Model_Name”] = Model_Name
```

INPUT:

```
ResoltML_Data
```

OUTPUT:

	Model_Name	SS_Score	RS_Score
0	GNB	NaN	NaN
1	BNB	NaN	NaN
2	SVC	NaN	NaN
3	LR	NaN	NaN
4	KNN	NaN	NaN
5	DTC	NaN	NaN
6	RFC	NaN	NaN

NAIVE BAYES CLASSIFICATION:

GaussianNB:

Input:

```
For i in range(0,2):
     gnb = GaussianNB()
    gnb.fit(train_test[i], y_train)
    y_pred = gnb.predict(train_test[i+2])
   gnb_csv = cross_val_score(estimator = gnb, X = train_test[i], y = y_train, cv = 5)
 print("GaussianNB Accuracy: ", accuracy_score(y_pred,y_test))
    print("GaussianNB Test Score: ", gnb.score(train_test[i+2], y_test))
    print("GaussianNB Train Score: ", gnb.score(train_test[i], y_train))
   print("GaussianNB Cross Validation Mean: ", gnb_csv.mean())
    print("GaussianNB Cross Validation Std: ", gnb_csv.std())
    print("----------------------------------------")
        if (i == 0):
        ResoltML_Data["SS_Score"][0] = accuracy_score(y_pred,y_test)
    else:
        ResoltML_Data["RS_Score"][0] = accuracy_score(y_pred,y_test)
```

OUTPUT:

GaussianNB Accuracy: 0.7719298245614035

GaussianNB Test Score: 0.7719298245614035

GaussianNB Train Score: 0.759765625

GaussianNB Cross Validation Mean: 0.7538930135160861

GaussianNB Cross Validation Std: 0.05600625999452157

GaussianNB Accuracy: 0.7719298245614035

GaussianNB Test Score: 0.7719298245614035

GaussianNB Train Score: 0.759765625

GaussianNB Cross Validation Mean: 0.7538930135160861

GaussianNB Cross Validation Std: 0.05600625999452157

BERNOULI NB:

INPUT:

```
For i in range(0,2):
    bnb = BernoulliNB()
   bnb.fit(train_test[i], y_train)
   y_pred_bnb = bnb.predict(train_test[i+2])
   bnb_cvs = cross_val_score(estimator = gnb, X = train_test[i], y = y_train, cv = 5)
 print("GaussianNB Accuracy: ", accuracy_score(y_pred_bnb,y_test))
   print("GaussianNB Test Score: ", bnb.score(train_test[i+2], y_test))
   print("GaussianNB Train Score: ", bnb.score(train_test[i], y_train))
   print("GaussianNB Cross Validation Mean: ", bnb_cvs.mean())
   print("GaussianNB Cross Validation Std: ", bnb_cvs.std())
   print("----------------------------------------")
    if (i == 0):
      ResoltML_Data["SS_Score"][1] = accuracy_score(y_pred_bnb,y_test)
   else:
      ResoltML_Data["RS_Score"][1] = accuracy_score(y_pred_bnb,y_test)
```

OUTPUT:

GaussianNB Accuracy: 0.7251461988304093

GaussianNB Test Score: 0.7251461988304093

GaussianNB Train Score: 0.732421875

GaussianNB Cross Validation Mean: 0.7538930135160861

GaussianNB Cross Validation Std: 0.05600625999452157

GaussianNB Accuracy: 0.7251461988304093

GaussianNB Test Score: 0.7251461988304093

GaussianNB Train Score: 0.73046875

GaussianNB Cross Validation Mean: 0.7538930135160861

GaussianNB Cross Validation Std: 0.05600625999452157

Support vector machine classification:

INPUT:

```
For i in range(0,2):
    svc = SVC()
  p_svc = [{'C':[1,2,3,4,5],'kernel':['linear']},
    {'C':[1,2,3,4,5],'kernel':['rbf'],'gamma':[1,0.5,0.1,0.01,0.001]},
    {'C':[1,2,3,4,5],'kernel':['poly'],'degree':[1,2,3,4,5,6,7],'gamma':[1,0.5,0.1,0.01,0.001]}]
  grid = GridSearchCV(estimator = svc, param_grid = p_svc, scoring = "accuracy", cv = 4)
  grid_search = grid.fit(train_test[i], y_train)
  y_pred_svc = grid_search.predict(train_test[i+2])
  best_parm_grid = grid_search.best_params_
  best_score_grid = grid_search.best_score_
  print("Best prameter of gridseach function: ", grid_search.best_params_)
  print("Best score of gridsearch function: ",grid_search.best_score_ )
  if (i == 0):
    ResoltML_Data["SS_Score"][2] = accuracy_score(y_pred_svc,y_test)
  else:
    ResoltML_Data["RS_Score"][2] = accuracy_score(y_pred_svc,y_test)
```

OUTPUT:

Best prameter of gridseach function: {‘C’: 5, ‘gamma’: 0.01, ‘kernel’: ‘rbf’}

Best score of gridsearch function: 0.787109375

Best prameter of gridseach function: {‘C’: 1, ‘degree’: 1, ‘gamma’: 0.5, ‘kernel’: ‘poly’}

Best score of gridsearch function: 0.78515625

Logistic Regression:

INPUT:

```
For i in range(0,2):
    logr = LogisticRegression(random_state = 0)
    p_lr = [{“penalty” : [“l1”,”l2”], “solver” : [“newton-cg”, “lbfgs”, “liblinear”, “sag”, “saga”],
          “multi_class” : [“auto”,”ovr”,”multinomial”]}]
    grid_lr = GridSearchCV(estimator = logr, param_grid = p_lr, scoring = “accuracy”, cv = 4)
    grid_search_lr = grid_lr.fit(train_test[i], y_train)
    y_pred_lr = grid_search.predict(train_test[i+2])
    best_parm_grid_lr = grid_search_lr.best_params_
    best_score_grid_lr = grid_search_lr.best_score_
print(“Best prameter of gridseach function: “, best_parm_grid_lr)
    print(“Best score of gridsearch function: “, best_score_grid_lr )
if (i == 0):
       ResoltML_Data[“SS_Score”][3] = accuracy_score(y_pred_lr,y_test)
    else:
       ResoltML_Data[“RS_Score”][3] = accuracy_score(y_pred_lr,y_test)
```

OUTPUT:

Best prameter of gridseach function: {‘multi_class’: ‘auto’, ‘penalty’: ‘l2’, ‘solver’: ‘liblinear’}

Best score of gridsearch function: 0.787109375

Best prameter of gridseach function: {‘multi_class’: ‘auto’, ‘penalty’: ‘l2’, ‘solver’: ‘newton-cg’}

Best score of gridsearch function: 0.787109375

K-Nearest Neighbour (KNN):

INPUT:

```
For i in range(0,2):
    knn_grid = KNeighborsClassifier()
p_knn = {“n_neighbors” : range(1,100), “weights” : [“uniform”, “distance”], “algorithm” : [“auto”, “ball_tree”, “kd_tree”, “brute”], “p” : [1,2]}
 grid_knn = GridSearchCV(estimator = knn_grid, param_grid = p_knn, scoring = “accuracy”, cv = 4)
    grid_knn_search = grid_knn.fit(train_test[i], y_train)
    y_pred_grid_knn = grid_knn.predict(train_test[i+2])
    best_parm_grid_knn = grid_knn_search.best_params_
    best_score_grid_knn = grid_knn_search.best_score_
   print(“GridSearch ile knn modelinin en iyi parametırları: “, best_parm_grid_knn)
    print(“GridSearch ile knn modelinin en iyi skoru: “,best_score_grid_knn)
  if (i == 0):
        ResoltML_Data[“SS_Score”][4] = accuracy_score(y_pred_grid_knn,y_test)
    else:
        ResoltML_Data[“RS_Score”][4] = accuracy_score(y_pred_grid_knn,y_test)
```

OUTPUT:

GridSearch ile knn modelinin en iyi parametırları: {‘algorithm’: ‘auto’, ‘n_neighbors’: 19, ‘p’: 2, ‘weights’: ‘distance’}

GridSearch ile knn modelinin en iyi skoru: 0.771484375

GridSearch ile knn modelinin en iyi parametırları: {‘algorithm’: ‘auto’, ‘n_neighbors’: 33, ‘p’: 2, ‘weights’: ‘uniform’}

GridSearch ile knn modelinin en iyi skoru: 0.779296875

Decision Tree Classification:

INPUT:

```
For i in range(0,2):
    dtr = DecisionTreeClassifier()
p_dtc = {“criterion” : [“gini”, “entropy”,”log_loss”], “splitter” : [“best”, “random”], “max_features” : [“auto”, “sqrt”, “log2”]}
  grid_dtc = GridSearchCV(estimator = dtr, param_grid = p_dtc, scoring = “accuracy”, cv = 4)
    grid_dtc_search = grid_dtc.fit(train_test[i], y_train)
    y_pred_grid_dtc = grid_dtc.predict(train_test[i+2])
best_parm_grid_dtc = grid_dtc_search.best_params_
    best_score_grid_dtc = grid_dtc_search.best_score_
    print(“GridSearch ile dtc modelinin en iyi parametırları: “, best_parm_grid_dtc)
    print(“GridSearch ile dtc modelinin en iyi skoru: “,best_score_grid_dtc)
   if (i == 0):
        ResoltML_Data[“SS_Score”][5] = accuracy_score(y_pred_grid_dtc,y_test)
    else:
        ResoltML_Data[“RS_Score”][5] = accuracy_score(y_pred_grid_dtc,y_test)
```

OUTPUT:

GridSearch ile dtc modelinin en iyi parametırları: {‘criterion’: ‘entropy’, ‘max_features’: ‘log2’, ‘splitter’: ‘random’}

GridSearch ile dtc modelinin en iyi skoru: 0.701171875

GridSearch ile dtc modelinin en iyi parametırları: {‘criterion’: ‘entropy’, ‘max_features’: ‘log2’, ‘splitter’: ‘best’}

GridSearch ile dtc modelinin en iyi skoru: 0.689453125

Random Forest Classification:

INPUT:

```
For i in range(0,2):
    rfc = RandomForestClassifier()
    p_rfc = {“n_estimators” : range(1,50), “criterion” : [“gini”, “entropy”,”log_loss”], “max_features” : [“sqrt”,”log2”, None],
            “class_weight” : [“balanced”, “balanced_subsample”]}
  grid_rfc = GridSearchCV(estimator = rfc, param_grid = p_rfc, scoring = “accuracy”, cv = 4)
    grid_rfc_search = grid_rfc.fit(train_test[i], y_train)
    y_pred_grid_rfc = grid_rfc.predict(train_test[i+2])
   best_parm_grid_rfc = grid_rfc_search.best_params_
    best_score_grid_rfc = grid_rfc_search.best_score_
print(“GridSearch ile rfc modelinin en iyi parametırları: “, best_parm_grid_rfc)
    print(“GridSearch ile rfc modelinin en iyi skoru: “,best_score_grid_rfc)
if (i == 0):
        ResoltML_Data[“SS_Score”][6] = accuracy_score(y_pred_grid_rfc,y_test)
    else:
        ResoltML_Data[“RS_Score”][6] = accuracy_score(y_pred_grid_rfc,y_test)
```

OUTPUT:

_GridSearch ile rfc modelinin en iyi parametırları: {‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘log2’, ‘n_estimators’: 32}

GridSearch ile rfc modelinin en iyi skoru: 0.783203125

GridSearch ile rfc modelinin en iyi parametırları: {‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘sqrt’, ‘n_estimators’: 26}

GridSearch ile rfc modelinin en iyi skoru: 0.7734375

Machine Learning Model Assessment:

INPUT:

ResoltML_Data

OUTPUT:

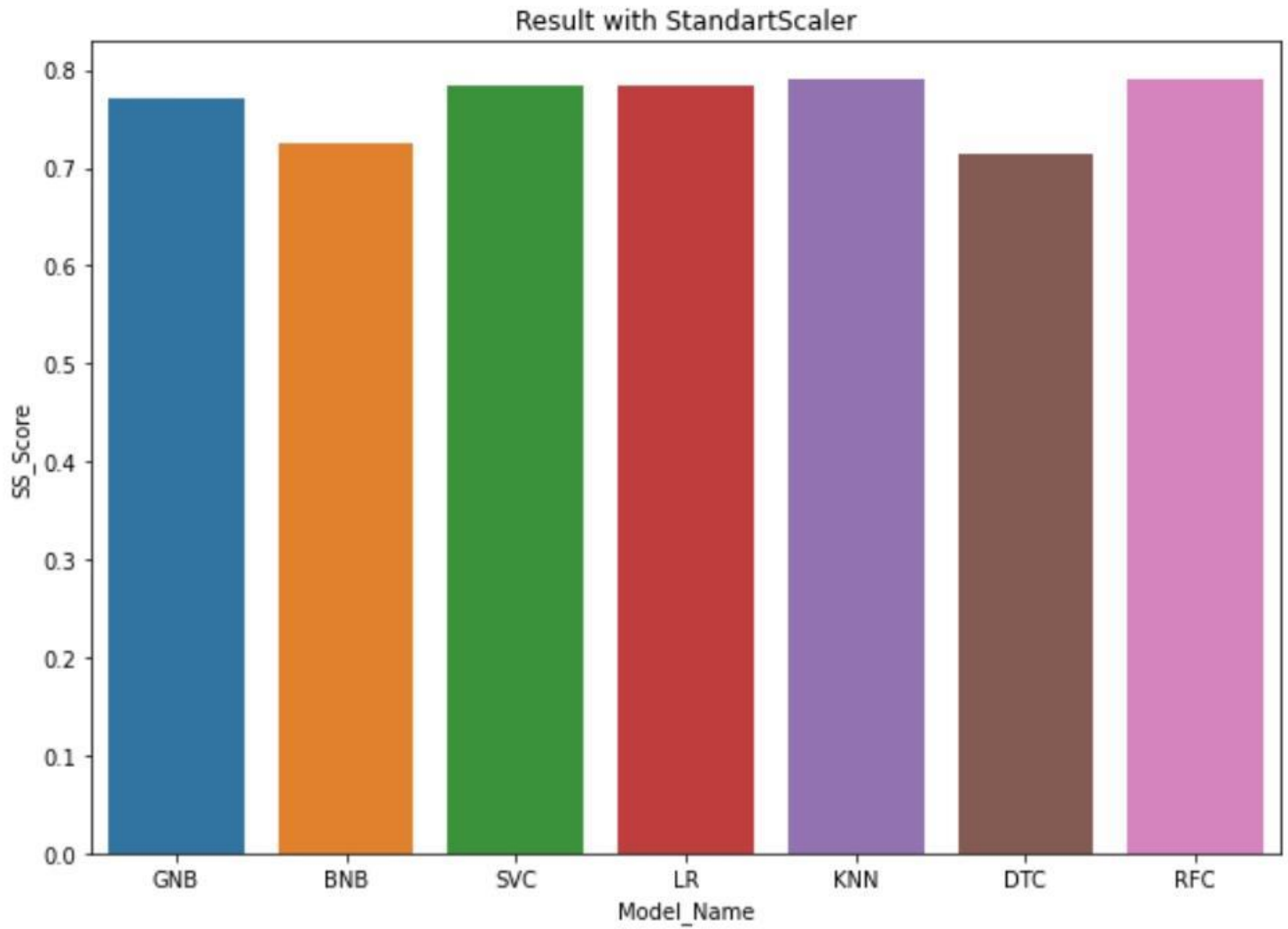
	Model_Name	SS_Score	RS_Score
0	GNB	0.77193	0.77193
1	BNB	0.725146	0.725146
2	SVC	0.783626	0.789474
3	LR	0.783626	0.789474
4	KNN	0.789474	0.789474
5	DTC	0.71345	0.672515
6	RFC	0.789474	0.766082

INPUT:

```
Plt.figure(figsize=(10,7))

sns.barplot(x=ResoltML_Data[“Model_Name”],
y=ResoltML_Data[“SS_Score”])

plt.xticks(rotation=0)

plt.xlabel(“Model_Name”)

plt.ylabel(“SS_Score”)

plt.title(“Result with StandartScaler”)
```

OUTPUT:

```
Text(0.5, 1.0, ‘Result with StandartScaler’)
```

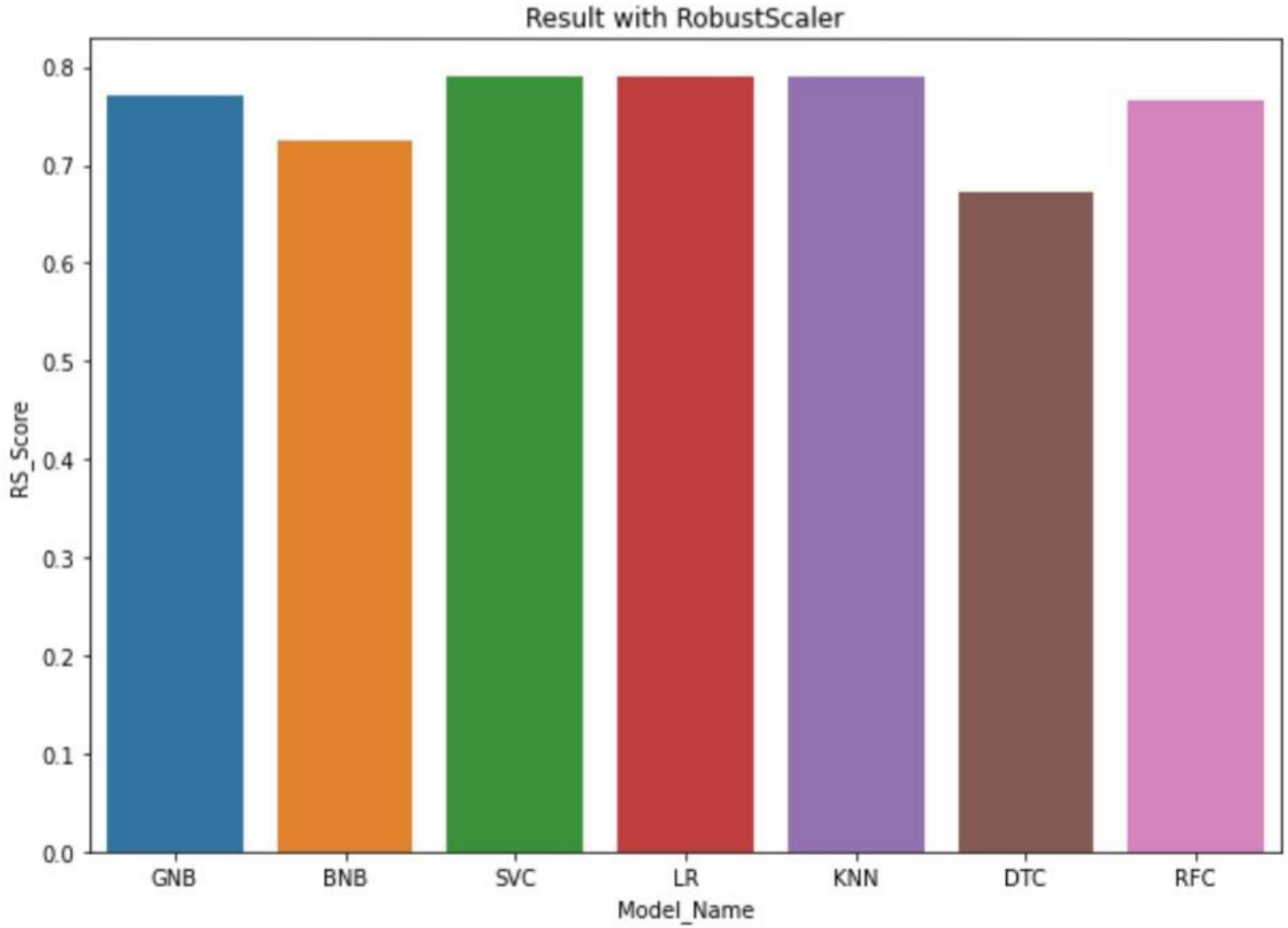

Result with StandartScaler
SS_Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
GNB
BNB
SVC
LR
KNN
DTC
RFC
Model_Name

INPUT:

```
Plt.figure(figsize=(10,7))

sns.barplot(x=ResoltML_Data[“Model_Name”],
y=ResoltML_Data[“RS_Score”])

plt.xticks(rotation=0)

plt.xlabel(“Model_Name”)

plt.ylabel(“RS_Score”)

plt.title(“Result with RobustScaler”)
```

OUTPUT:

```
Text(0.5, 1.0, ‘Result with RobustScaler’)
```


Result with RobustScaler
RS_Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
GNB
BNB
SVC
LR
KNN
DTC
RFC
Model_Name

Conclusion:

☆In conclusion, the development of an AI-based diabetes prediction system is a significant and innovative project with the potential to positively impact healthcare and improve patient outcomes. This project involves a range of critical steps, from data collection and preprocessing to model training, evaluation, and deployment. Here are the key takeaways:

- **Data is Fundamental:** The quality and quantity of data used in the project are of utmost importance. Gathering a comprehensive dataset and preprocessing it meticulously is the foundation of a successful diabetes prediction system.

- **Feature Engineering**: Careful feature selection and engineering are crucial for enhancing the predictive power of the model. It's essential to focus on relevant features and transform them appropriately.
- **Model Selection**: Choosing the right machine learning or deep learning model is essential. The model should be well-suited to the problem and dataset.
- **Evaluation and Validation:** Thoroughly evaluate the model's performance using appropriate metrics and validation techniques. Collaborate with domain experts to ensure the system's accuracy and reliability.
- **Evaluation and Validation**: Thoroughly evaluate the model's performance using appropriate metrics and validation techniques. Collaborate with domain experts to ensure the system's accuracy and reliability.

- **Ethical and Privacy Considerations**: Developing a healthcare-related AI system comes with ethical and privacy responsibilities. Safeguarding patient data and ensuring the model's transparency and fairness are paramount.
- **Visualization and Interpretability**: Incorporating meaningful visualizations and interpretability tools can help users, including healthcare professionals and patients, understand the model's predictions and trust the system.
- **User Education and Collaboration**: Actively involve healthcare professionals, provide guidance on interpreting model results, and encourage users to consult healthcare experts for diagnosis and treatment.
- **Continuous Improvement:** Healthcare is a dynamic field, and continuous monitoring and updates are necessary to keep the system accurate and up-to-date.

- **Regulatory Compliance**: Ensure that the project complies with healthcare regulations and ethical standards. Collaborate with legal experts to navigate the regulatory landscape
- **Impact and Future Directions**: A successful AI-based diabetes prediction system can have a profound impact on public health. Consider future enhancements like risk stratification, personalized recommendations, and integration with wearable devices.

☆In summary, the AI-based diabetes prediction system project is a complex endeavor that requires a multidisciplinary approach involving data science, healthcare expertise, and ethical considerations. When executed diligently, it has the potential to improve diabetes diagnosis and management, ultimately enhancing the quality of patient care.