**AI BASED DIABETES PREDICTION SYSTEM**

**PHASE 4 Submission Document**

**Phase 4 : Development Part 2**

**Topic : Continue building the AI based diabetes prediction system model by feature engineering, model training, and evaluation.**

****

**Introduction**

* Diabetes is a chronic disease that directly affects the pancreas, and the body is incapable of producing insulin. It is mainly responsible for maintaining the blood glucose level.
* Diabetes is generally characterized by either the body not making enough insulin or being unable to use the insulin that is made as effectively as needed.
* To use machine learning classification methods, that is, decision tree, SVM, Random Forest, Logistic Regression, KNN, and various ensemble techniques, to determine which algorithm produces the best prediction results.
* In this paper, we have employed machine learning and explainable AI techniques to detect diabetes.

**Abstract**

Artificial intelligence (AI)-based diabetes prediction systems are a promising new approach to early detection and prevention of diabetes. These systems use machine learning to analyze a person's medical history and other risk factors to predict their risk of developing diabetes. This information can then be used to develop personalized risk assessments and treatment plans.

**DatasetLink:**[**https://www.kaggle.com/datasets/mathchi/diabetes-data-set**](https://www.kaggle.com/datasets/mathchi/diabetes-data-set)

**Diabetes dataset**



**Chats**

**Benefits of using AI-based diabetes prediction systems:**

* **Early detection:** AI-based systems can help to identify people at risk of developing diabetes early on, when the disease is most treatable.
* **Prevention:** By identifying people at risk, AI-based systems can help people to take steps to prevent the development of diabetes in the first place.
* **Personalized care:**AI-based systems can be used to develop personalized risk assessments and treatment plans for people with diabetes or prediabetes.
* **Improved health outcomes:** AI-based systems can help people with diabetes to better manage their condition and improve their overall health outcomes.

**AI-based diabetes prediction systems typically consist of the following modules:**

* **Data collection and preparation:** This module collects and prepares data from a variety of sources, such as electronic health records, wearable devices, and patient surveys. The data is then cleaned and preprocessed to ensure that it is in a format that can be used by the machine learning models.
* **Feature engineering:**This module extracts features from the data that are relevant to predicting diabetes risk. For example, features might include age, sex, BMI, blood pressure, and family history of diabetes.
* **Machine learning modeling:**This module trains a machine learning model to predict diabetes risk based on the extracted features. A variety of machine learning algorithms can be used for this task, such as logistic regression, support vector machines, and random forests.
* **Model evaluation:** This module evaluates the performance of the trained machine learning model on a held-out test set. This helps to ensure that the model is generalizable to new data and that it can accurately predict diabetes risk.
* **Risk assessment and treatment planning:** This module uses the trained machine learning model to develop personalized risk assessments and treatment plans for patients. This information can then be used by patients and their healthcare providers to make informed decisions about diabetes prevention and management.

**Feature engineering**

Feature engineering is the process of transforming raw data into features that are more informative and predictive for a machine learning model. In the context of AI-based diabetes prediction systems, feature engineering can be used to:

* Create new features that are more directly related to diabetes risk, such as blood sugar variability or insulin resistance.
* Combine existing features in new and meaningful ways, such as calculating a patient's risk score based on their age, BMI, and blood pressure.
* Transform features into a format that is more compatible with the machine learning algorithm being used.

Some specific examples of feature engineering techniques that can be used for AI-based diabetes prediction include:

* Binning: Binning is the process of discretizing continuous features. This can be useful for features that have a wide range of values, such as blood sugar levels.
* One-hot encoding: One-hot encoding is the process of converting categorical features into a set of binary features. This can be useful for features with a large number of categories, such as sex or ethnicity.
* Feature scaling: Feature scaling is the process of normalizing the values of all features to a common scale. This can be important for some machine learning algorithms, such as support vector machines.
* Creating new features: New features can be created by combining existing features in new and meaningful ways. For example, a new feature could be created to represent a patient's blood sugar variability by calculating the standard deviation of their blood sugar levels over time.

Feature engineering is an important part of developing any AI-based system, including AI-based diabetes prediction systems. By carefully engineering the features that are used to train the machine learning model, it is possible to improve the accuracy and performance of the system.

Here are some additional tips for feature engineering AI-based diabetes prediction systems:

* Use domain knowledge to identify features that are likely to be predictive of diabetes risk.
* Use a variety of feature engineering techniques to create a rich set of features.
* Evaluate the performance of different feature engineering techniques on a held-out test set.

**Model training**

To train an AI-based diabetes prediction system model, you will need to:

1. Collect a dataset of diabetes patients and healthy individuals. The dataset should include features that are known to be associated with diabetes, such as age, sex, weight, height, blood pressure, blood glucose levels, and family history.
2. Prepare the dataset for machine learning. This may involve cleaning the data, removing outliers, and scaling the features.
3. Choose a machine learning algorithm. There are many different machine learning algorithms that can be used for diabetes prediction, such as support vector machines (SVMs), random forests, and logistic regression.
4. Train the model on the dataset. This involves feeding the dataset to the machine learning algorithm and allowing it to learn the patterns associated with diabetes.
5. Evaluate the model on a held-out test set. This will give you an idea of how well the model will perform on new data.
6. Deploy the model. Once you are satisfied with the model's performance, you can deploy it to production so that it can be used to predict diabetes risk in new individuals.

**Here are some additional tips for training an AI-based diabetes prediction system model:**

* Use a large and diverse dataset. The larger and more diverse your dataset, the better the model will be able to learn the patterns associated with diabetes.
* Use feature engineering to create new features that may be more predictive of diabetes. For example, you could create a feature that represents the patient's body mass index (BMI).
* Use a cross-validation procedure to evaluate the model. This involves splitting the dataset into multiple folds and training the model on each fold. The model's performance is then evaluated on the folds that it was not trained on. This helps to reduce overfitting and get a more accurate estimate of the model's performance.
* Use a variety of machine learning algorithms and compare their performance. Choose the algorithm that performs best on your dataset.
* Consider using explainable AI (XAI) techniques to understand how the model is making predictions. This can help you to identify potential biases in the model and to build trust in the model's predictions.

Once you have trained a diabetes prediction system model, you can use it to predict the risk of diabetes in new individuals. This information can be used to develop personalized prevention and treatment plans.

**Evaluation**

The evaluation of AI-based diabetes prediction systems is an important step in ensuring that they are reliable and accurate. There are a number of different metrics that can be used to evaluate these systems, including:

* Accuracy: This is the percentage of predictions that the system makes correctly.
* Precision: This is the percentage of positive predictions that are correct.
* Recall: This is the percentage of actual positives that the system predicts correctly.
* F1 score: This is a harmonic mean of precision and recall, and it is often used to evaluate the performance of binary classifiers.
* AUC-ROC curve: This is a curve that shows the trade-off between sensitivity and specificity. The area under the curve (AUC) is a measure of the overall performance of the classifier.

In addition to these metrics, it is also important to consider the following factors when evaluating an AI-based diabetes prediction system:

* The size and diversity of the dataset: The model should be trained on a large and diverse dataset of diabetes patients and healthy individuals. This will help to ensure that the model is able to generalize to new data.
* The evaluation methodology: The model should be evaluated on a held-out test set. This will give you an idea of how well the model will perform on new data.
* The explainability of the model: It is important to be able to explain how the model is making predictions. This will help you to identify potential biases in the model and to build trust in the model's predictions.

Once you have evaluated the AI-based diabetes prediction system, you can decide whether or not to deploy it to production. If you decide to deploy the system, it is important to monitor its performance over time and to update it as needed.

Here are some additional tips for evaluating an AI-based diabetes prediction system:

* Use multiple metrics to evaluate the system. This will give you a more complete picture of the system's performance.
* Compare the system's performance to other diabetes prediction systems. This will help you to determine whether or not the system is state-of-the-art.
* Consider the clinical significance of the system's predictions. For example, a system that has high accuracy but low precision may not be clinically useful, as it may generate many false positives.
* Involve clinicians in the evaluation process. This will help to ensure that the system meets the needs of the users.

By carefully evaluating an AI-based diabetes prediction system, you can ensure that it is reliable and accurate, and that it can be used to improve the care of patients with diabetes.

**Program**

**AI Based Diabetes Prediction System**

# Import libraries

**import** numpy as np # for linear algebra

**import** pandas as pd # for data processing, CSV file I/O (e.g. pd.read\_csv)

**import** seaborn as sns # for data visualization

**import** matplotlib.pyplot as plt # to plot data visualization charts

**from** collections **import** Counter

**import** os

# Modeling Libraries

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score, precision\_score

**from** sklearn.preprocessing **import** QuantileTransformer

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.ensemble **import** RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier

**from** sklearn.model\_selection **import** GridSearchCV, cross\_val\_score, StratifiedKFold, learning\_curve, train\_test\_split

**from** sklearn.svm **import** SVC

**Importing the Dataset**

# Importing the dataset from Kaggle

data = pd.read\_csv("../input/pima-indians-diabetes-database/diabetes.csv")

# First step is getting familiar with the structure of the dataset

data.info()

**Output**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Pregnancies 768 non-null int64

1 Glucose 768 non-null int64

2 BloodPressure 768 non-null int64

3 SkinThickness 768 non-null int64

4 Insulin 768 non-null int64

5 BMI 768 non-null float64

6 DiabetesPedigreeFunction 768 non-null float64

7 Age 768 non-null int64

8 Outcome 768 non-null int64

dtypes: float64(2), int64(7)

memory usage: 54.1 KB

**Code**

# Showing the top 5 rows of the dataset

data.head()

**Output**

****

**Filling the Missing Values** **Code**

# Exploring the missing values in the diabetes dataset

data.isnull().sum()

**Output**

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

**Code**

# Replacing 0 values with the mean of that column

# Replacing 0 values of Glucose

data['Glucose'] = data['Glucose'].replace(0, data['Glucose'].median())

# Filling 0 values of Blood Pressure

data['BloodPressure'] = data['BloodPressure'].replace(0, data['BloodPressure'].median())

# Replacing 0 values in BMI

data['BMI'] = data['BMI'].replace(0, data['BMI'].mean())

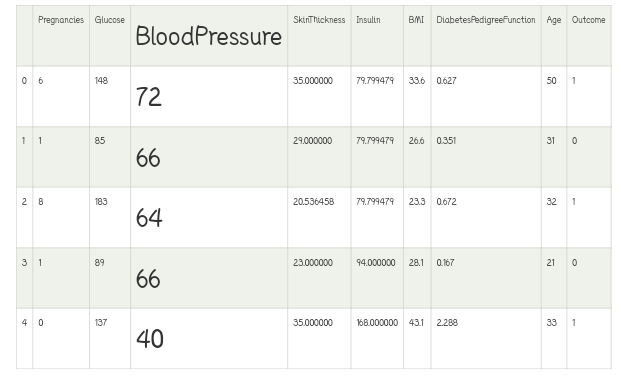
# Replacing the missing values of Insulin and SkinThickness

data['SkinThickness'] = data['SkinThickness'].replace(0, data['SkinThickness'].mean())

data['Insulin'] = data['Insulin'].replace(0, data['Insulin'].mean())

data.head()

**Output**

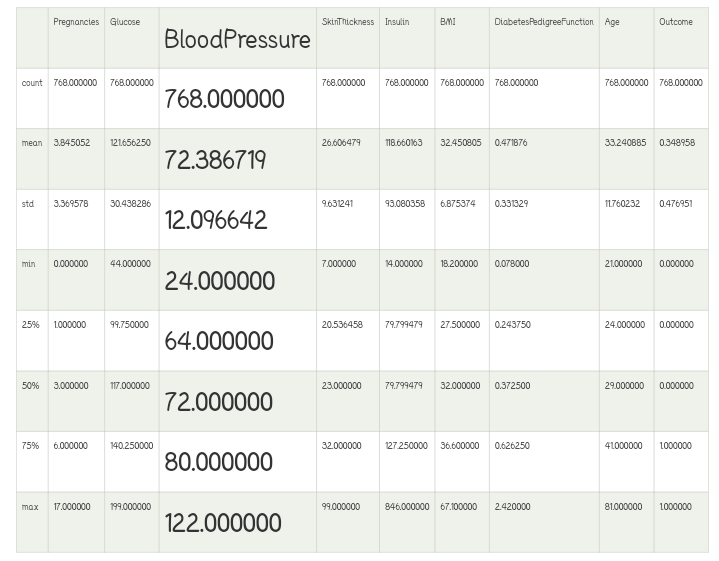
****

**Code**

# Reviewing the dataset statistics

data.describe()

**Output**

****

**Code**

# Defining the Logistic Regression model and its parameters

model = LogisticRegression(solver ='liblinear')

solver\_list = ['liblinear']

penalty\_type = ['l2']

c\_values = [200, 100, 10, 1.0, 0.01]

# Defining the grid search

grid\_lr = dict(solver = solver\_list, penalty = penalty\_type, C = c\_values)  cross\_val = StratifiedKFold(n\_splits = 100, random\_state = 10, shuffle = True)

grid\_search\_cv = GridSearchCV(estimator = model, param\_grid = grid\_lr, cv = cross\_val, scoring = 'accuracy', error\_score = 0)

lr\_result = grid\_search\_cv.fit(X\_train, Y\_train)

# Result of Hyper Parameters of Logistic Regression

analyze\_grid(lr\_result)

**Output**

Tuned hyperparameters: {'C': 200, 'penalty': 'l2', 'solver': 'liblinear'}

Accuracy Score: 0.7715000000000001

Mean: 0.7715000000000001, Std: 0.16556796187668676 \* 2, Params: {'C': 200, 'penalty': 'l2', 'solver': 'liblinear'}

The classification Report:

Mean: 0.7715000000000001, Std: 0.16556796187668676 \* 2, Params: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}

The classification Report:

Mean: 0.7675, Std: 0.16961353129983467 \* 2, Params: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}

The classification Report:

Mean: 0.7675, Std: 0.17224619008848932 \* 2, Params: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}

The classification Report:

Mean: 0.711, Std: 0.1888888562091475 \* 2, Params: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}

The classification Report:

precision recall f1-score support

0 0.78 0.88 0.83 201

1 0.70 0.53 0.61 107

accuracy 0.76 308

macro avg 0.74 0.71 0.72 308

weighted avg 0.75 0.76 0.75 308

**Pregnancy** **Code**

# Exploring Pregnancy and target variables together

  plt.figure(figsize = (10, 8))

# Plotting density function graph of the pregnancies and the target variable

kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 1], color = "Red", shade = True)

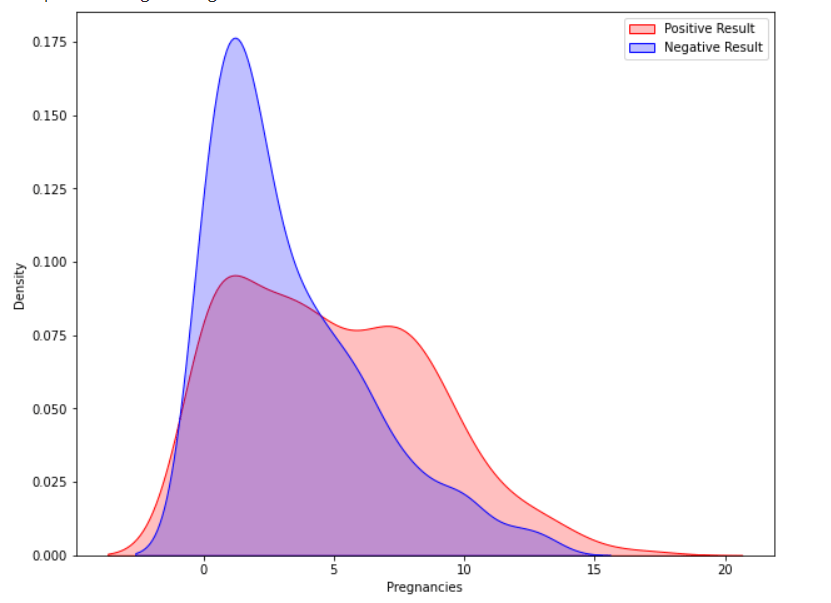
kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 0], ax = kde, color = "Blue", shade= True)

kde.set\_xlabel("Pregnancies")

kde.set\_ylabel("Density")

kde.legend(["Positive Result", "Negative Result"])

**Output**



**Glucose code**

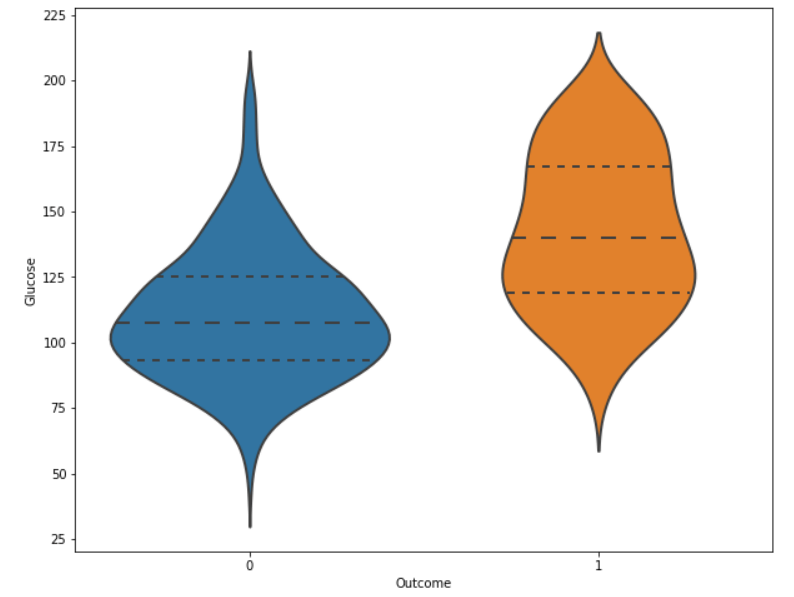
# Exploring the Glucose and the Target variables together

plt.figure(figsize = (10, 8))

sns.violinplot(data = data, x = "Outcome", y = "Glucose",

               split = True, inner = "quart", linewidth = 2)

**Output**



**Code**

**# Histogram and density graphs of all variables were accessed.**

**fig, ax = plt.subplots(4,2, figsize=(16,16))**

**sns.distplot(df.Age, bins = 20, ax=ax[0,0])**

**sns.distplot(df.Pregnancies, bins = 20, ax=ax[0,1])**

**sns.distplot(df.Glucose, bins = 20, ax=ax[1,0])**

**sns.distplot(df.BloodPressure, bins = 20, ax=ax[1,1])**

**sns.distplot(df.SkinThickness, bins = 20, ax=ax[2,0])**

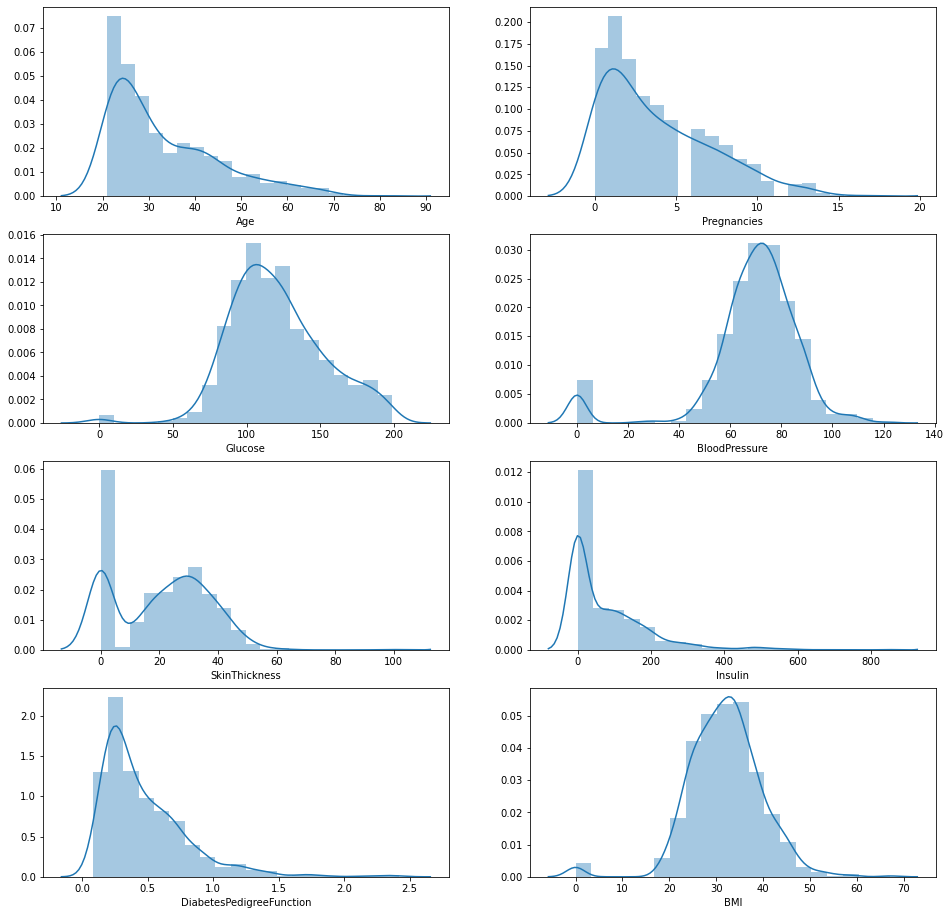
**sns.distplot(df.Insulin, bins = 20, ax=ax[2,1])**

**sns.distplot(df.DiabetesPedigreeFunction, bins = 20, ax=ax[3,0])**

**sns.distplot(df.BMI, bins = 20, ax=ax[3,1])**

**Output**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7f77b83d5950>**

****

**Future for diabetes prediction system**

* Demographic features: Age, sex, race/ethnicity, and family history of diabetes are all important risk factors for diabetes.
* Lifestyle factors: Body mass index (BMI), waist circumference, physical activity level, and diet are all lifestyle factors that can influence the risk of developing diabetes.
* Clinical data: Blood glucose levels, blood pressure, cholesterol levels, and hemoglobin A1c (HbA1c) are all important clinical data points that can be used to predict diabetes risk.

**Conclusions**

* AI-based diabetes prediction systems can be used to identify people at risk of developing diabetes with high accuracy.
* AI-based diabetes prediction systems can be used to develop personalized risk assessments and treatment plans for people with diabetes or prediabetes.
* AI-based diabetes prediction systems can help to improve the quality of life for people with diabetes by helping them to better manage their condition and avoid complications.
* Artificial intelligence (AI)-based diabetes prediction systems have the potential to revolutionize the way we diagnose and manage diabetes. By analyzing large amounts of data, AI systems can identify patterns and correlations that would be difficult or impossible for humans to detect. This information can be used to predict which individuals are at high risk of developing diabetes, so that they can take steps to prevent the disease.