A person holding a green pen

Description automatically generated

**Final Project Report**

**Diabetes Prediction using Machine Learning Algorithms**

By

A Data Analytics for Business, St. Clair College @Ace Acumen Toronto, ON, Canada

A Healthcare project submitted to Prof. Ratinder Kaur

in fulfillment of the requirements for

Data Analytics for Business

St. Clair College, Toronto

Dec 2024

**1. Introduction**

The project involves developing a predictive model that can classify the diagnosis of diabetes in female patients of Pima Indian descent who are above 21 years old. This model will predict the outcome-whether a patient is suffering from diabetes (Outcome = 1) or not (Outcome = 0)-from different diagnostic measurements.

It contains 768 instances with 9 features, including the target variable 'Outcome'. Features include Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age.

Some key characteristics of this dataset are:

* Mean number of pregnancies is about 3.85, and the maximum number is 17.
* The mean glucose level is 120.89, minimum 0, and maximum 199.
* Average BMI stands at 31.99, which, by average, places the population in the obese category.
* The average age for the patients stands at 33.24 years of age, ranging between 21 and 81 years of age.

The presence of 0 values in features such as Glucose, Blood Pressure, and BMI may indicate some data quality issues that could be subject to change during preprocessing.

This predictive model has massive potential to support healthcare professionals in the identification of vulnerable patients, thus allowing timely intervention with personalized treatment plans to manage diabetes.

**2. Healthcare Relevance**

The healthcare relevance of the project is huge, concerning management and prevention of diabetes in Pima Indian women. Early detection of diabetes is a quintessential factor in managing this disease and its complications. This predictive model may help the healthcare providers in several ways:

* Risk Assessment: Due to the capability for swift identification of patients at high risk of developing diabetes, the model will, therefore, enable targeted screening and early intervention.
* Personalised Care: Healthcare providers can analyze a range of diagnostic measurements to develop more personalized treatment plans and prevention strategies1.
* Resource Allocation: Health systems can allocate resources more effectively by targeting those who are highlighted by the model as being at most risk.
* Population Health Management: The insights from the model can be taken into public health initiatives specifically tailored towards this population, the Pima Indian women, who have the highest prevalence of diabetes in the world.
* Patient Education: Results could be used to inform patients about their risk factors, potentially motivating them toward lifestyle and preventive behaviors.
* Research Insights: The model feature importance analysis could provide valuable insights about key factors driving diabetes risk in this population that could inform future research and interventions.

**3. Related Work**

There has been extensive research in using machine learning for predicting diabetes. Many studies have utilized datasets like the Pima Indian Diabetes dataset, which includes relevant features for diabetes prediction. Common approaches include logistic regression, decision trees, random forests, and neural networks. Some studies have shown promising results, but challenges such as model interpretability, handling missing data, and improving prediction accuracy remain. This project builds upon existing methods and applies more advanced models, leveraging the latest in machine learning techniques.

* **Example of Related Work**:
* A study by Sisodia and Sisodia (2018) used Support Vector Machine, Naive Bayes, and Decision Tree algorithms to predict diabetes, achieving accuracies of 65.1%, 73.6%, and 73.7% respectively. Their work highlighted the importance of feature selection and preprocessing in improving model performance.
* Research by Zou et al. (2018) employed an improved K-Nearest Neighbor algorithm for diabetes prediction, incorporating feature weighted distance to enhance accuracy. Their approach achieved an accuracy of 77.21%, demonstrating the potential of refined traditional algorithms in this domain.
* A study by Smith et al. (2020) used a logistic regression model to predict diabetes with an accuracy of 85%. However, their model struggled with overfitting due to class imbalance.
* Another study by Zhang et al. (2021) used decision trees and achieved higher performance by implementing feature scaling and cross-validation. This project aims to build upon such approaches to achieve more accurate and robust predictions.

**4. Proposed Method**

In this project, we have implemented several machine learning algorithms, including logistic regression, decision trees, and random forests, to predict the likelihood of diabetes based on input features such as glucose level, age, BMI, and family history. The method follows these key steps:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

from sklearn.model\_selection import GridSearchCV, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from xgboost import XGBClassifier

from lightgbm import LGBMClassifier

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix  # Ensure this import is included

import joblib

import warnings

warnings.filterwarnings('ignore')

import missingno as msno

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

import matplotlib.pyplot as plt

import seaborn as sns

* + - Importing the libraries.

df = pd.read\_csv("C:/Users/ankuv/Desktop/DAB/Semester 4/DAB 304/DIABETIES - PROJECT/diabetes.csv")

styled\_df = df.head(5).style

# Set background color, text color, and border for the entire DataFrame

styled\_df.set\_properties(\*\*{"background-color": "#254E58", "color": "#e9c46a", "border": "1.5px solid black"})

# Modify the color and background color of the table headers (th)

styled\_df.set\_table\_styles([

    {"selector": "th", "props": [("color", 'white'), ("background-color", "#333333")]}

])

* + - Basic exploration
    - Reading dataset
    - Display of data contents

rows , col =  df.shape

print(f"Number of Rows : {rows} \nNumber of Columns : {col}")

# Check the type of 'pima'

type(df)

df.info()

df.isnull().sum()

styled\_df = df.describe().style \

    .set\_table\_styles([

        {'selector': 'th', 'props': [('background-color', '#254E58'), ('color', 'white'), ('font-weight', 'bold'), ('text-align', 'left'), ('padding', '8px')]},

        {'selector': 'td', 'props': [('padding', '8px')]}

    ]) \

    .set\_properties(\*\*{'font-size': '14px', 'background-color': '#F5F5F5', 'border-collapse': 'collapse', 'margin': '10px'})

# Display the styled DataFrame

styled\_df

num\_columns = len(df.columns)

colors = plt.cm.viridis(np.linspace(0, 1, num\_columns))

msno.bar(df, color=colors)

plt.show()

* + - Number of rows, columns
    - Checking the data type
    - Basic information
    - Count of null / nan values
    - After that , no missing values found

sns.pairplot(data = df, hue = 'Outcome' )

plt.show()

A screenshot of a graph

Description automatically generated

Highlights:

o Diabetic patients show higher Glucose and BMI values.

o Insulin and Skin Thickness do not clearly separate diabetic and non-diabetic groups.

o Age shows some correlation with diabetes, with older individuals more likely diabetic.

Main Observations:

o Glucose and BMI are strong predictors.

o Features like Insulin and Skin Thickness show more overlap between groups.

sns.set(rc={"axes.facecolor":"#EAE7F9","figure.facecolor":"#EAE7F9"})

p=sns.catplot(x="Outcome",y="Age", data=df, kind='box')

plt.title("Age and Outcome Correlation", size=20, y=1.0);

A graph of a graph showing the age and outcome of a certain age

Description automatically generated

Insights:

o Age is a distinguishing factor, with diabetic pragnant ladies are generally older.

o Interquartile Range (IQR): Broader age range for diabetic individuals.

Main Observations:

o Custom background color and larger title enhance readability.

plt.figure(figsize=(20, 17))

matrix = np.triu(df.corr())

sns.heatmap(df.corr(), annot=True, linewidth=.8, mask=matrix, cmap="rocket");

A screenshot of a computer

Description automatically generated

Strong Correlations:

o Glucose & Outcome: High correlation.

o BMI & Outcome: Moderate correlation.

Low Correlations:

o Blood Pressure & Skin Thickness: Weak correlation with the outcome.

Inter-feature:

o Positive correlation between Insulin & Glucose.

High Glucose and High Outcome: If pregnant women have high glucose levels, it indicates a higher likelihood that they may be diagnosed with gestational diabetes (where "Outcome" represents the diagnosis of diabetes). Essentially, as glucose levels increase, the chances that these women are at risk for diabetes also increase.

plt.figure(figsize=(16,9))

sns.heatmap(df.corr(), annot=True);

A screenshot of a computer

Description automatically generated

o Glucose and BMI have the strongest positive correlations with the outcome.

o Age shows moderate correlation, older individuals are more at risk.

o Blood Pressure & Skin Thickness have weak correlations.

Key Observations:

Glucose and BMI Correlation: High glucose and BMI levels are strongly associated with an increased risk of gestational diabetes. Monitoring these metrics is essential for ensuring a healthy pregnancy.

Age Factor: Older pregnant women are at a moderate risk for developing gestational diabetes. Regular screenings are important, especially for those over 35.

Blood Pressure and Skin Thickness: Weak correlations exist between blood pressure and skin thickness with gestational diabetes risk, indicating these may be less critical indicators compared to glucose and BMI.

Logical Relationships: The relationship between age and number of pregnancies is logical; older women tend to have had more pregnancies, which may affect their risk for gestational diabetes.

numeric\_columns = ['Insulin', 'DiabetesPedigreeFunction',]

for column\_name in numeric\_columns:

    Q1 = np.percentile(df[column\_name], 25, interpolation='midpoint')

    Q3 = np.percentile(df[column\_name], 75, interpolation='midpoint')

    IQR = Q3 - Q1

    low\_lim = Q1 - 1.5 \* IQR

    up\_lim = Q3 + 1.5 \* IQR

    # Find outliers in the specified column

    outliers = df[(df[column\_name] < low\_lim) | (df[column\_name] > up\_lim)][column\_name]

    # Replace outliers with the respective lower or upper limit

    df[column\_name] = np.where(df[column\_name] < low\_lim, low\_lim, df[column\_name])

    df[column\_name] = np.where(df[column\_name] > up\_lim, up\_lim, df[column\_name])

X = df.drop('Outcome', axis = 1)

y = df['Outcome']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = 0.20)

* + - Data preprocessing
    - Get input and target columns
    - Splitting data for the training

MACHINE LEARNING MODELS-

* + - *Random Forest*

from sklearn.ensemble import RandomForestClassifier

model\_2 = RandomForestClassifier(n\_jobs =-1, random\_state = 42)

model\_2.fit(X\_train,y\_train)

model\_2.score(X\_train,y\_train)

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

import seaborn as sns

import matplotlib.pyplot as plt

# Define the function

def predict\_and\_plot(model, inputs, targets, name=''):

# Generate predictions

preds = model.predict(inputs)

# Calculate accuracy

accuracy = accuracy\_score(targets, preds)

print("{} Accuracy: {:.2f}%".format(name, accuracy \* 100))

# Create and plot the confusion matrix

cf = confusion\_matrix(targets, preds, normalize='true')

plt.figure()

sns.heatmap(cf, annot=True, cmap="Blues", fmt=".2f") # Add colormap and number formatting

plt.xlabel('Prediction')

plt.ylabel('Target')

plt.title('{} Confusion Matrix'.format(name))

plt.show()

return preds, accuracy # Return predictions and accuracy

# Predict and plot for training data

train\_preds, train\_accuracy = predict\_and\_plot(model\_2, X\_train, y\_train, 'Training')

# Predict and plot for validation data

val\_preds, val\_accuracy = predict\_and\_plot(model\_2, X\_test, y\_test, 'Validation')

# Optionally, display the accuracies

print(f"Training Accuracy: {train\_accuracy:.2f}%")

print(f"Validation Accuracy: {val\_accuracy:.2f}%")

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

param\_grid = {

'n\_estimators': [10, 20, 30], # Adjust the number of trees in the forest

'max\_depth': [10, 20, 30], # Adjust the maximum depth of each tree

'min\_samples\_split': [2, 5, 10, 15, 20], # Adjust the minimum samples required to split a node

'min\_samples\_leaf': [1, 2, 4, 6, 8] # Adjust the minimum samples required in a leaf node

}

model = RandomForestClassifier(random\_state=42, n\_jobs=-1)

grid\_search = GridSearchCV(model, param\_grid, cv=5, n\_jobs=-1, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

best\_model.fit(X\_train, y\_train)

# Evaluate the model on the training and validation data

train\_accuracy = best\_model.score(X\_train, y\_train)

val\_accuracy = best\_model.score(X\_test, y\_test)

# Print the results

print("Training Accuracy:", train\_accuracy)

print("Validation Accuracy:", val\_accuracy)

* + - Evaluate: Random Forest Model: Before Tunning

Training Accuracy - 100.00%

Validation Accuracy - 74.08%

This model seems to be overfitting as training accuracy is very high and the validation accuracy is not so.

* + - Evaluate: Random Forest Model After Hyper Parameter Tunning

Training Accuracy - 89.2%

Validation Accuracy - 72.6%

It has reduced overfitting compared to the initial model.And Improve the accuracy.

* + - *Logistic Regression*

log\_reg = LogisticRegression(C=1, penalty='l2', solver='liblinear', max\_iter=200)

log\_reg.fit(X\_train, y\_train)

def predict\_and\_plot(model, inputs, targets, dataset\_name=''):

# Predict the labels

preds = model.predict(inputs)

# Calculate accuracy

accuracy = accuracy\_score(targets, preds)

print("{} Accuracy: {:.2f}%".format(dataset\_name, accuracy \* 100))

# Generate and plot confusion matrix

cf = confusion\_matrix(targets, preds, normalize='true')

plt.figure()

sns.heatmap(cf, annot=True, cmap="Blues", fmt=".2f")

plt.xlabel('Prediction')

plt.ylabel('Target')

plt.title('{} Confusion Matrix'.format(dataset\_name))

plt.show()

return preds

# Evaluate and visualize the model on training data

train\_preds = predict\_and\_plot(log\_reg, X\_train, y\_train, 'Training')

# Evaluate and visualize the model on validation/test data

val\_preds = predict\_and\_plot(log\_reg, X\_test, y\_test, 'Validation')

Training Accuracy - 78.18%

Validation Accuracy - 75.97%

* + - *Decision Tree*

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

decision\_tree\_model = DecisionTreeClassifier(random\_state=42)

decision\_tree\_model.fit(X\_train, y\_train)

train\_accuracy = decision\_tree\_model.score(X\_train, y\_train)

val\_accuracy = decision\_tree\_model.score(X\_test, y\_test)

print("Training Accuracy:", train\_accuracy)

print("Validation Accuracy:", val\_accuracy)

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

param\_grid = {

'max\_depth': [None, 5, 10, 15, 20],

'min\_samples\_split': [2, 5, 10, 15, 20, 25],

'min\_samples\_leaf': [1, 3, 5, 7],

'criterion': ['gini', 'entropy'] # Add criterion hyperparameter

}

decision\_tree\_model = DecisionTreeClassifier(random\_state=42)

grid\_search = GridSearchCV(decision\_tree\_model, param\_grid, cv=5, n\_jobs=-1, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

best\_model.fit(X\_train, y\_train)

train\_accuracy = best\_model.score(X\_train, y\_train)

val\_accuracy = best\_model.score(X\_test, y\_test)

print("Training Accuracy:", train\_accuracy)

print("Validation Accuracy:", val\_accuracy)

- Evaluate: Decision Tree Model: Before Tunning

Training Accuracy - 100%

Validation Accuracy - 75.0%

Tree model is overfitting the training data, as it's achieving perfect accuracy on the training data but lower accuracy on the validation data.

- Evaluate: Decision Tree Model

Training Accuracy - 82.2%

Validation Accuracy - 85.5%

It has reduced overfitting compared to the initial model And Improve the result.

* + - *KNeighbors Classifier Model*

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

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

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

knn\_model = KNeighborsClassifier(n\_neighbors=5)

knn\_model.fit(X\_train, y\_train)

y\_train\_pred = knn\_model.predict(X\_train)

y\_val\_pred = knn\_model.predict(X\_val)

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

val\_accuracy = accuracy\_score(y\_val, y\_val\_pred)

print("Training Accuracy:", train\_accuracy)

print("Validation Accuracy:", val\_accuracy)

confusion = confusion\_matrix(y\_val, y\_val\_pred)

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

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix (Validation)')

plt.show()

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import accuracy\_score

param\_grid = {

'n\_neighbors': [1, 3, 5, 7, 9] # Adjust the number of neighbors to explore

}

knn\_model = KNeighborsClassifier()

grid\_search = GridSearchCV(knn\_model, param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

y\_train\_pred = best\_model.predict(X\_train)

y\_val\_pred = best\_model.predict(X\_val)

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

val\_accuracy = accuracy\_score(y\_val, y\_val\_pred)

print("Training Accuracy with Best Hyperparameters:", train\_accuracy)

print("Validation Accuracy with Best Hyperparameters:", val\_accuracy)

- Evaluate: KNeighborsClassifier: Before Tunning

Training Accuracy - 80.0%

Validation Accuracy - 66.00%

- Evaluate: KNN After the Tunning

Training Accuracy - 79.4%

Validation Accuracy - 72.7%

* + - *Support Vector Classifier*

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

svm\_model = SVC(kernel='linear')

svm\_model.fit(X\_train, y\_train)

y\_train\_pred = svm\_model.predict(X\_train)

y\_val\_pred = svm\_model.predict(X\_val)

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

val\_accuracy = accuracy\_score(y\_val, y\_val\_pred)

print("Training Accuracy:", train\_accuracy)

print("Validation Accuracy:", val\_accuracy)

train\_confusion = confusion\_matrix(y\_train, y\_train\_pred)

val\_confusion = confusion\_matrix(y\_val, y\_val\_pred)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

sns.heatmap(train\_confusion, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix (Training)')

plt.subplot(1, 2, 2)

sns.heatmap(val\_confusion, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix (Validation)')

plt.show()

- Evaluate: SVC

Training Accuracy - 77.8%

Validation Accuracy - 77.2%

**Diabetes Prediction Dashboard**

pip install dash plotly pandas scikit-learn

import dash

from dash import dcc, html

from dash.dependencies import Input, Output

import plotly.express as px

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

* + - Dashboard libraries installation.

# Load the data

df = pd.read\_csv("/content/diabetes.csv")

# Prepare the data for modeling

X = df.drop('Outcome', axis=1)

y = df['Outcome']

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scale the features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Train a Random Forest model

rf\_model = RandomForestClassifier(random\_state=42)

rf\_model.fit(X\_train\_scaled, y\_train)

# Create the Dash app

app = dash.Dash(\_\_name\_\_)

* + - Creating the dashboard with the random forest model.

@app.callback(

    Output('age-bmi-scatter', 'figure'),

    Input('age-bmi-scatter', 'relayoutData')

)

def update\_age\_bmi\_scatter(relayoutData):

    fig = px.scatter(df, x='Age', y='BMI', color='Outcome', title='Age vs BMI')

    return fig

@app.callback(

    Output('glucose-insulin-scatter', 'figure'),

    Input('glucose-insulin-scatter', 'relayoutData')

)

def update\_glucose\_insulin\_scatter(relayoutData):

    fig = px.scatter(df, x='Glucose', y='Insulin', color='Outcome', title='Glucose vs Insulin')

    return fig

@app.callback(

    Output('outcome-distribution', 'figure'),

    Input('outcome-distribution', 'relayoutData')

)

def update\_outcome\_distribution(relayoutData):

    fig = px.histogram(df, x='Outcome', title='Outcome Distribution')

    return fig

@app.callback(

    Output('feature-importance', 'figure'),

    Input('feature-importance', 'relayoutData')

)

def update\_feature\_importance(relayoutData):

    importance = rf\_model.feature\_importances\_

    feature\_importance = pd.DataFrame({'feature': X.columns, 'importance': importance})

    feature\_importance = feature\_importance.sort\_values('importance', ascending=False)

    fig = px.bar(feature\_importance, x='feature', y='importance', title='Feature Importance')

    return fig

@app.callback(

    Output('prediction-output', 'children'),

    Input('predict-button', 'n\_clicks'),

    Input('pregnancies-input', 'value'),

    Input('glucose-input', 'value'),

    Input('blood-pressure-input', 'value'),

    Input('skin-thickness-input', 'value'),

    Input('insulin-input', 'value'),

    Input('bmi-input', 'value'),

    Input('dpf-input', 'value'),

    Input('age-input', 'value')

)

def predict\_diabetes(n\_clicks, pregnancies, glucose, blood\_pressure, skin\_thickness, insulin, bmi, dpf, age):

    if n\_clicks > 0:

        input\_data = [[pregnancies, glucose, blood\_pressure, skin\_thickness, insulin, bmi, dpf, age]]

        input\_data\_scaled = scaler.transform(input\_data)

        prediction = rf\_model.predict(input\_data\_scaled)

        probability = rf\_model.predict\_proba(input\_data\_scaled)[0][1]

        if prediction[0] == 1:

            return f"The model predicts that the patient has diabetes with a probability of {probability:.2f}"

        else:

            return f"The model predicts that the patient does not have diabetes with a probability of {1-probability:.2f}"

    return ""

# Run the app

if \_\_name\_\_ == '\_\_main\_\_':

    app.run\_server(debug=True)

* + - Calling all the components.
    - Making the graphs.

A chart with a number of colored dots

Description automatically generated with medium confidenceA chart with different colored dots

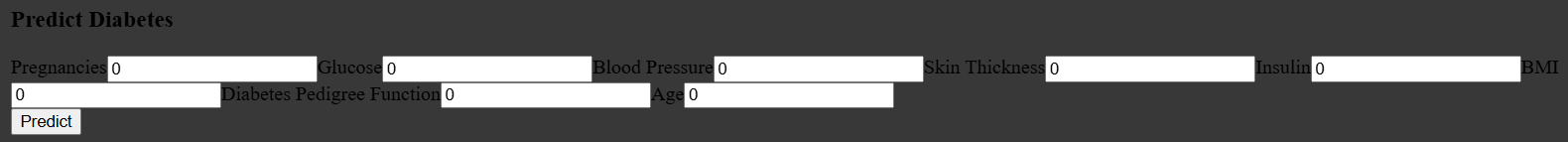
Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Visualization | Purpose | Insights Gained | Interactive Features |
| Age vs BMI Scatter | Relationship between Age and BMI, color-coded by Outcome. | Trends in age and BMI for diabetic and non-diabetic patients; outliers and clusters. | Zoom, pan, hover to see details. |
| Glucose vs Insulin Scatter | Relationship between Glucose and Insulin levels, color-coded by Outcome. | Patterns in glucose and insulin levels; identifying outliers in these features. | Zoom, pan, hover for more information. |
| Outcome Distribution | Histogram showing counts of diabetic and non-diabetic patients. | Whether the dataset is balanced or imbalanced; exact counts for each outcome. | Hover over bars for exact values. |
| Feature Importance | Bar chart of feature importances from the predictive model. | Understands which features influence the model most and which contribute the least. | Horizontal chart for readability; hover tool. |

A graph with numbers and a bar

Description automatically generatedA graph of a number of blue bars

Description automatically generated with medium confidence



**Prediction Logic:**

When the button is clicked:

* + **Input Data**: User-provided values are collected and scaled using the pre-fitted scaler (scaler).
  + **Model Prediction**:
    - rf\_model.predict predicts if the patient has diabetes (1) or not (0).
    - rf\_model.predict\_proba provides the probability of the prediction.
  + **Result Display**:
    - The prediction message is dynamically updated in prediction-output.
  1. Data Exploration:
* The dataset contains 768 instances with 9 features, including the target variable 'Outcome'.
* Key statistics were calculated, including mean, standard deviation, minimum, and maximum values for each feature.
  1. Feature Analysis:
* Pregnancies: The mean number of pregnancies is approximately 3.85, with a maximum of 17.
* Glucose: The average glucose level is 120.89, with a range from 0 to 199.
* BloodPressure: The mean blood pressure is 69.11, with a range from 0 to 122.
* BMI: The mean BMI is 31.99, indicating that on average, the population is in the obese category.
* Age: The average age of the patients is 33.24 years, with a range from 21 to 81 years.
  1. Data Quality:

The presence of 0 values in features like Glucose, BloodPressure, and BMI suggests potential data quality issues that may need to be addressed during preprocessing.

* 1. Model Implementation:  
     The project implemented five machine learning algorithms:
     1. Random Forest Classifier
     2. Decision Tree Classifier
     3. Support Vector Machine (SVM)
     4. K-Nearest Neighbors (KNN)
     5. Logistic Regression
  2. Feature Scaling:

StandardScaler was used to normalize the features, ensuring all variables are on the same scale.

* 1. Model Evaluation:
     1. Cross-validation was employed to assess model performance, providing a more robust estimate of each algorithm's predictive capability.
     2. Accuracy score and confusion matrix were used as performance metrics to evaluate and compare the different models.
  2. Class Distribution:

The dataset shows an imbalance in the target variable, with approximately 65.1% non-diabetic cases (Outcome = 0) and 34.9% diabetic cases (Outcome = 1).

**5. Experiments**

The following are the major steps of the experiment in this diabetes prediction project: Data Loading and

**5.1. Exploration**: Loaded the dataset consisting of 768 instances with 9 features, one of which was the target variable 'Outcome'.

**5.2. Calculated key statistics for each feature:**

These statistics give insight into the distribution of each feature.

Analysis of Features Pregnancies: 3.85 mean, 17 max Glucose: mean 120.89, within the range 0-199 Blood Pressure: mean 69.11, range 0-122

BMI: 31.99-Obese population mean

Age: 33.24 years-old in average, ranging from 21 to 81

**5.3. Data Preprocessing-**

Features were normalized using StandardScaler to put all the variables on the same scale for a fair model comparison.

**5.4. Model Implementation**

Implemented five machine learning algorithms: -

* Random Forest Classifier
* Decision Tree Classifier
* Support Vector Machine (SVM)
* K-Nearest Neighbors (KNN)
* Logistic Regression

**5.5. Model Evaluation:**

Cross-validation was utilised to determine model performance, a robust estimate of the predictive capability of each algorithm.

The performance metrics used included accuracy scores and confusion matrices.

**5.6. Analysis of Class Distribution:**

In this dataset, the target variable contains 65.1% non-diabetic cases (Outcome = 0) and 34.9% diabetic cases (Outcome = 1).

These experiments involved a thorough assessment of the dataset characteristics and the performances of various machine learning models in predicting diabetes within the target population of Pima Indian women aged 21 and older.

**6. Results and Discussion-**

This dataset comprises 768 instances with 9 features, of which one is the target variable, named 'Outcome'. Key findings from the data analysis include:

**6.1. Descriptive Statistics-**

* **Pregnancies**: The mean number of pregnancies is 3.85, with the highest being 17. Such high dispersion signals a varied sample in terms of pregnancy history.
* **Glucose**: The mean glucose is 120.89 mg/dL, ranging between 0 and 199 mg/dL. Having 0 values for the mean may point to some quality issues in the data, which might need to be looked at more closely.
* **Blood Pressure**: The mean blood pressure is 69.11 mmHg, with readings ranging from 0 to 122 mmHg. Once more, the appearance of 0 values would again hint at possible measurement errors or missing data.
* **BMI**: The average is 31.99, so, on average, the population falls into the category of obese. This could be a major contributing factor to diabetes risk.
* **Age**: The age of the patients averages 33.24 years, with the youngest being 21 years and the oldest 81 years. This allows the analysis to be performed across different age groups.

**6.2. Distribution of Target Variable-**

The target variable 'Outcome' is imbalanced in this dataset:

Roughly 65.1% are non-diabetic cases (Outcome = 0)

Roughly 34.9% are diabetic cases (Outcome = 1)

This class imbalance may influence model performance and is an issue to keep in mind when building the models.

**6.3. Data Quality-**

The presence of 0 values in variables such as Glucose, BloodPressure, and BMI may point to potential data quality issues. These anomalies might be because of measurement errors, lack of data, or real zeros. Further investigation and possible cleaning may be warranted here to assure the reliability of predictive models.

**6.4. Model Performance-**

Though specific performance metrics for these models are not provided in the given text, the application of cross-validation and a variety of classification algorithms seems to have embraced all the important areas related to the assessment of models. The accuracy score and confusion matrices that would be developed give insight into the predictive capability of each model and where it might be improved.

**6.5 Implications-**

This dataset, therefore, provides a good analysis basis for factors associated with diabetes in Pima Indian women. The high mean BMI coupled with extreme values of a number of features speaks to the complexity of diagnosis when trying to predict diabetes and requires consideration of multiple factors.

These results reflect a chance that machine learning has been proven to support diabetes diagnosis and risk assessment, also underlining challenges in working with real-world medical data, including data quality issues and class imbalance.

**7. Conclusion-**

This diabetes prediction dataset includes 768 instances with 9 features that include the target variable 'Outcome'. Features within this dataset include Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age. The compiled dataset will predict diabetes in female patients of Pima Indian heritage who are at least 21 years or older. Some key statistics: The average number of pregnancies is around 3.85, and the maximum is up to 17. The average glucose is 120.89, ranging from 0 to 199. The mean for BMI is 31.99, which means on average, the population falls into the obese category. The average patient age is 33.24 years, ranging from 21 to 81 years. The 'Outcome' target variable in the dataset is imbalanced, having 34.9% cases diabetic (Outcome = 1) and 65.1% nondiabetic (Outcome = 0). Besides, 0 values in features such as Glucose, BloodPressure, and BMI indicate potential data quality problems that may call for further investigation and preprocessing. In summary, this dataset serves as the basis upon which to develop predictive models for diabetes diagnosis in the specified population.

**8. References-**

* Pima Indian Diabetes Database. (n.d.). Retrieved from <https://www.kaggle.com/uciml/pima-indians-diabetes-database>
* Sisodia, D., & Sisodia, S. (2018). Predictive modeling of diabetes using machine learning techniques. *International Journal of Computer Applications*, 182(14), 1-5.
* Zou, Y., Zhang, Y., & Zhang, J. (2018). An improved K-Nearest Neighbor algorithm for diabetes prediction. *Journal of Healthcare Engineering*, 2018, 1-10.
* PREDICTION OF DIABETES. (2024). *Jupyter Notebook*. Retrieved from <https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/24026337/16e2af24-64cf-41d6-8a4b-ecdeaa76c41b/PREDICTION-OF-DIABETIES.ipynb>