Term Project Report

Full Unit – Final Report

Diabetes Prediction using Machine Learning

Usama Raheem

A report submitted in part fulfilment of the term project of

BSc (CS-329) Artificial Intelligence

Supervisor: Dr. Naveed Anwar Butt

Department of Computer Science

Hafiz Hayat, University of Gujrat

October 05, 2022

Declaration

This report has been prepared on the basis of my research work with guidance of my superior colleagues. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 873

Student Name: **Usama Raheem**

Student Roll No: **19011519-002**

Student Section: **CS-A**

Student Batch: **CS-19**

Date of Submission: **27th March 2022 1:15 PM EST**

Contents

Abstract 3

Project Specification. 4

Chapter 1: Introduction 5

1.1 Overview 5

1.2 Data Description 5

1.3 Data Exploration (EDA) 6

Chapter 2: Modeling .....

Chapter 3: Cross Validation Scores on Best Parameters .....

Chapter 4: Deployment Diabetes Prediction .....

Abstract

This document contain report based on Diabetes Prediction dataset trained on well-known machine learning technique and models. The Author carefully predicts the probability of any individual's having diabetes disease purely based on the that particular individual body features. The author successfully trains various machine learning models and decisively compare the predicted outcome and choose the best parameter to find the optimize predicted scores. At the end, the author deployed the best model based on Accuracy successfully deployed on Web by using Flask Framework.

Project Specification.

The main objective of this term project is to Predict the Probability of an individual having Diabetes Disease based on diagnostic measures. In this project, Author uses dataset consists of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has, their BMI, insulin level, age etc. The Main task the author specify here is to build a machine learning model to accurately predict whether or not the patient in the dataset have the diabetes or not.

Chapter 1: **Introduction**

The project report contains the steps and procedure of an end-to-end example of solving a real-world problem using Data Science. The author will be using Machine Learning to predict whether a person has diabetes or not, specifically based on information about the patient such as blood pressure, body mass index (BMI), age. This term project report walks through the various stages of the data science workflow. In particular the Report has following sections

- Overview
- Data Description
- Data Exploration

1.1 Overview

The data was collected and made available by "National Institute of Diabetes and Digestive and Kidney Diseases". Several constraints were placed on the selection of these instances from a larger database. The diabetes dataset was easily available on Kaggle to play with and attain some useful insights from the dataset.

1.2 Data Description

In data Description the author has some columns in dataset which were given below:

Pregnancies: Number of times pregnant

Glucose: Plasma glucose concentration 2 hours in an oral glucose tolerance test

Blood Pressure: Diastolic blood pressure (mm Hg)

Skin Thickness: Triceps skin fold thickness (mm)

Insulin: 2-Hour serum insulin (mu U/ml)

BMI: Body mass index (weight in kg/(height in m)^2)

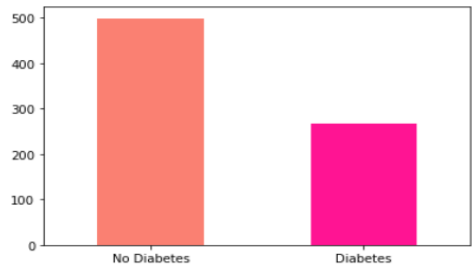
DiabetesPedigreeFunction: It provided some data on diabetes mellitus history in relatives and the genetic relationship of those relatives to the patient.

Age: Age (years)

Outcome: Class variable (0 or 1) 268 of 768 are 1, the others are 0

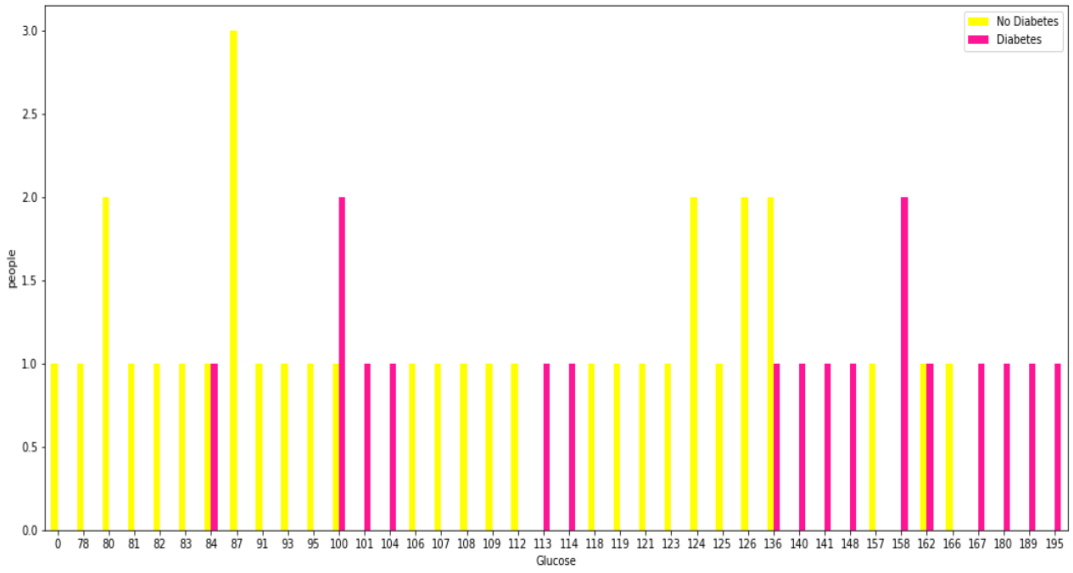
1.3 Data Exploration (EDA)

```
[11]: data["Outcome"].value_counts().plot(kind="bar",color=["salmon","deeppink"])
      plt.xticks(np.arange(2), ('No Diabetes', 'Diabetes'),rotation=0);
```

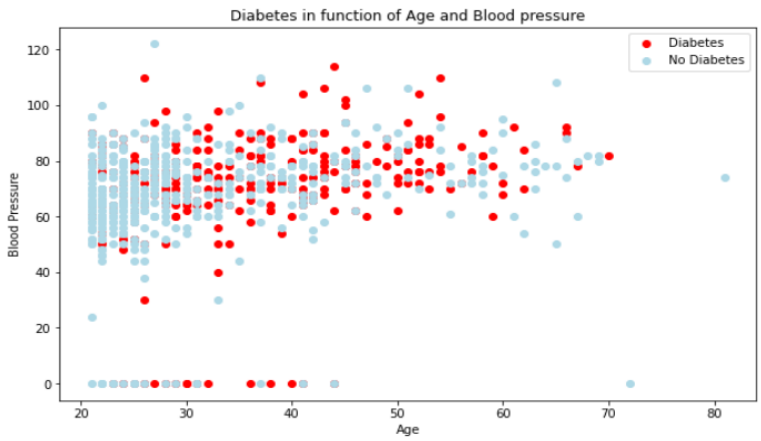


Comparison Glucose with Outcome

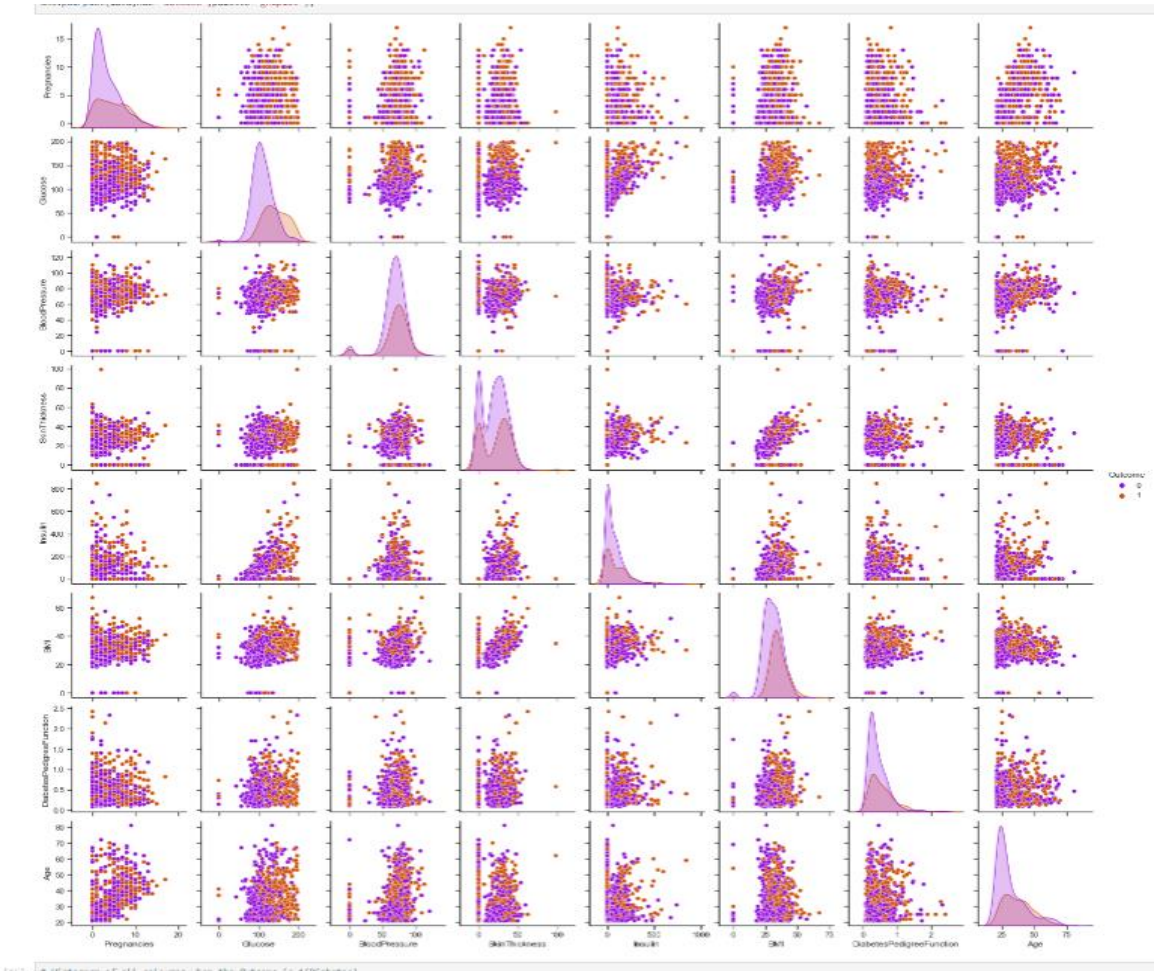
```
[13]: # Comparing Glucose with the Outcome
      pd.crosstab(data.Glucose[::15],data.Outcome).plot(kind="bar",figsize=(18,8),color=["yellow","deeppink"])
      plt.ylabel("people");
      plt.xticks(rotation=0);
      plt.legend(['No Diabetes', 'Diabetes']);
```



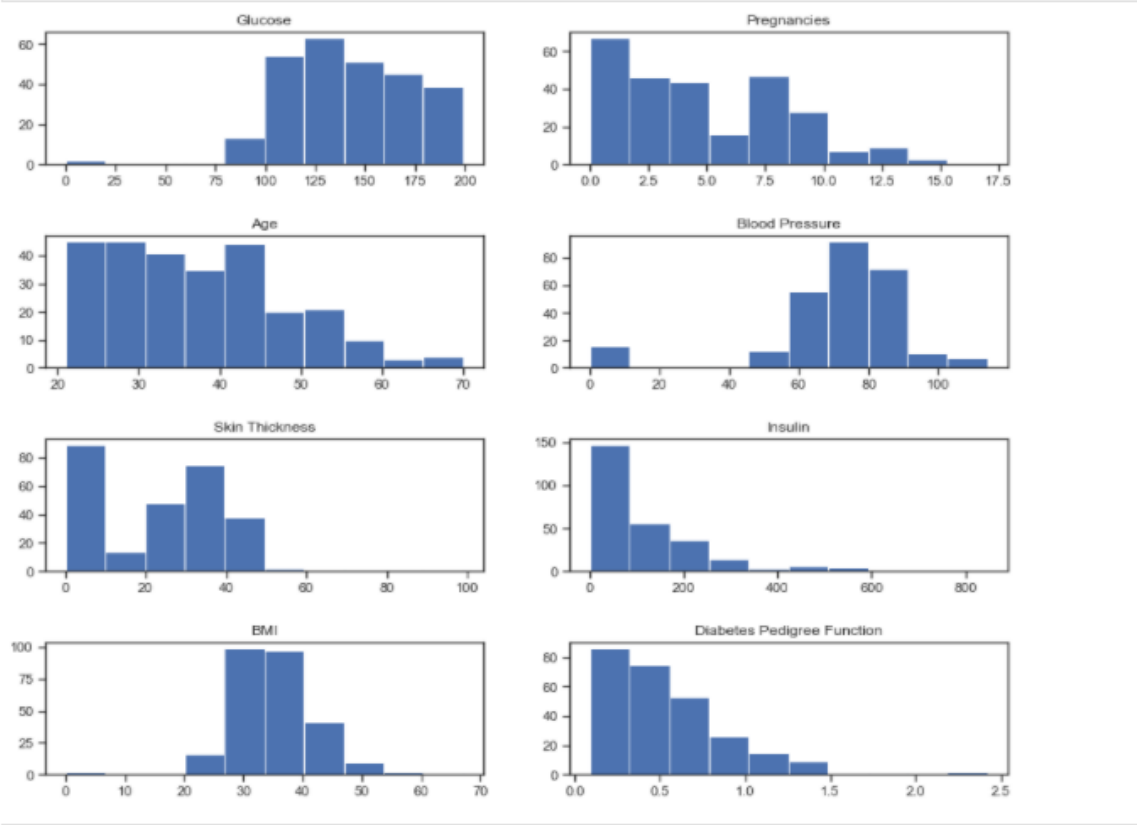
Diabetes in function of Age and Blood Pressure



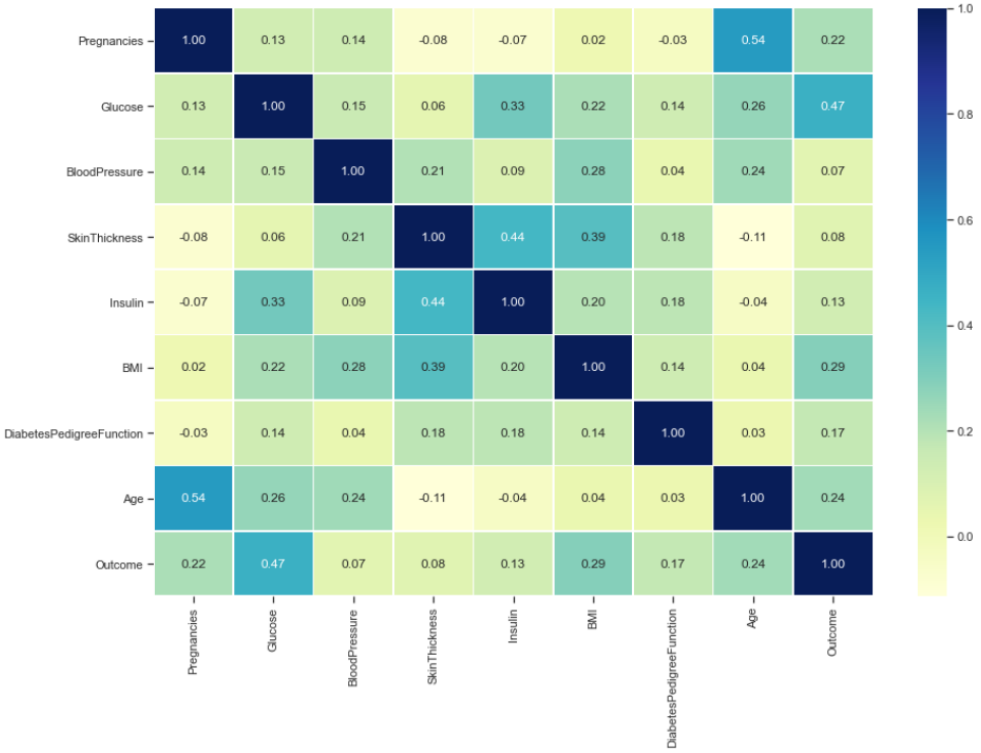
Diabetic and Non-Diabetic Pair Plot using Seaborn



Histogram of All columns when the outcome is 1 (Diabetic)



Correlation Matrix Visualization

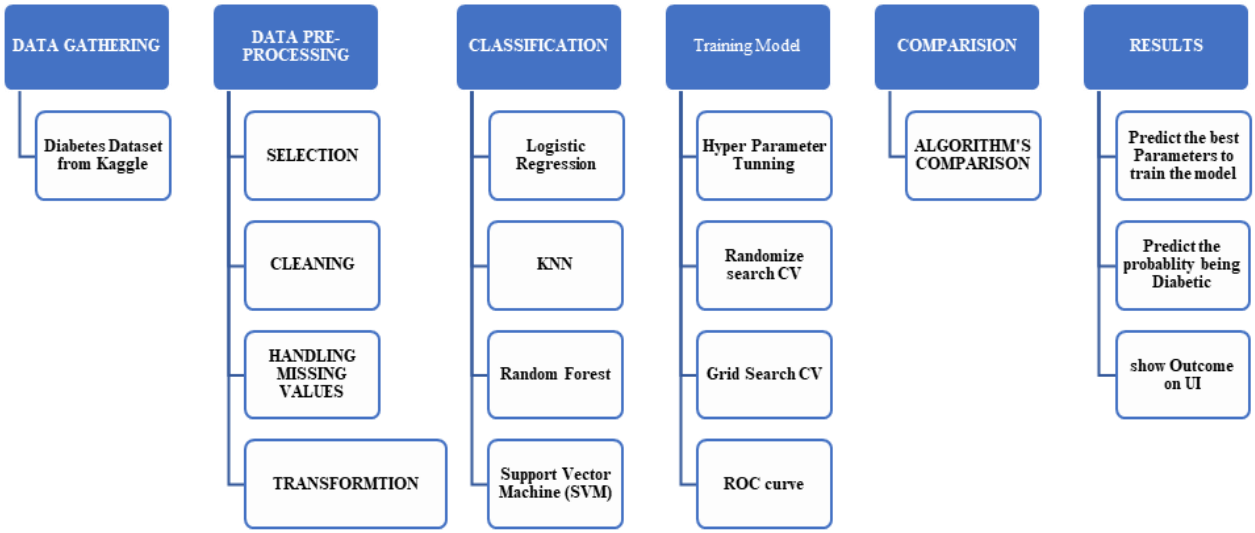


Dataset Features

[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.128621	0.141399	-0.081009	-0.074356	0.017469	-0.034065	0.543904	0.221272
Glucose	0.128621	1.000000	0.152718	0.058120	0.330836	0.220957	0.136886	0.262783	0.466143
BloodPressure	0.141399	0.152718	1.000000	0.207390	0.089003	0.281825	0.041300	0.239699	0.065135
SkinThickness	-0.081009	0.058120	0.207390	1.000000	0.437635	0.392867	0.184412	-0.113312	0.075426
Insulin	-0.074356	0.330836	0.089003	0.437635	1.000000	0.197744	0.184728	-0.042985	0.129973
BMI	0.017469	0.220957	0.281825	0.392867	0.197744	1.000000	0.140546	0.036031	0.292612
DiabetesPedigreeFunction	-0.034065	0.136886	0.041300	0.184412	0.184728	0.140546	1.000000	0.033044	0.173478
Age	0.543904	0.262783	0.239699	-0.113312	-0.042985	0.036031	0.033044	1.000000	0.237725
Outcome	0.221272	0.466143	0.065135	0.075426	0.129973	0.292612	0.173478	0.237725	1.000000

Chapter 2: **Modelling**



Classification

1. Logistic Regression

```
## Build an model (Logistic Regression)
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state=0)
log_reg.fit(X_train,y_train);
## Evaluating the model
log_reg = log_reg.score(X_test,y_test)
```

2. KNN

```
## Build an model (KNN)
knn = KNeighborsClassifier()
knn.fit(X_train,y_train);
## Evaluating the model
knn = knn.score(X_test,y_test)
```

3. Random Forest Classifier

```
## Build an model (Random forest classifier)
clf= RandomForestClassifier()
clf.fit(X_train,y_train);
## Evaluating the model
clf = clf.score(X_test,y_test)
```

4. Support Vector Machine (SVM)

```
## Build an model (Support Vector Machine)
svm = SVC()
svm.fit(X_train,y_train)
svm = svm.score(X_test,y_test)
```

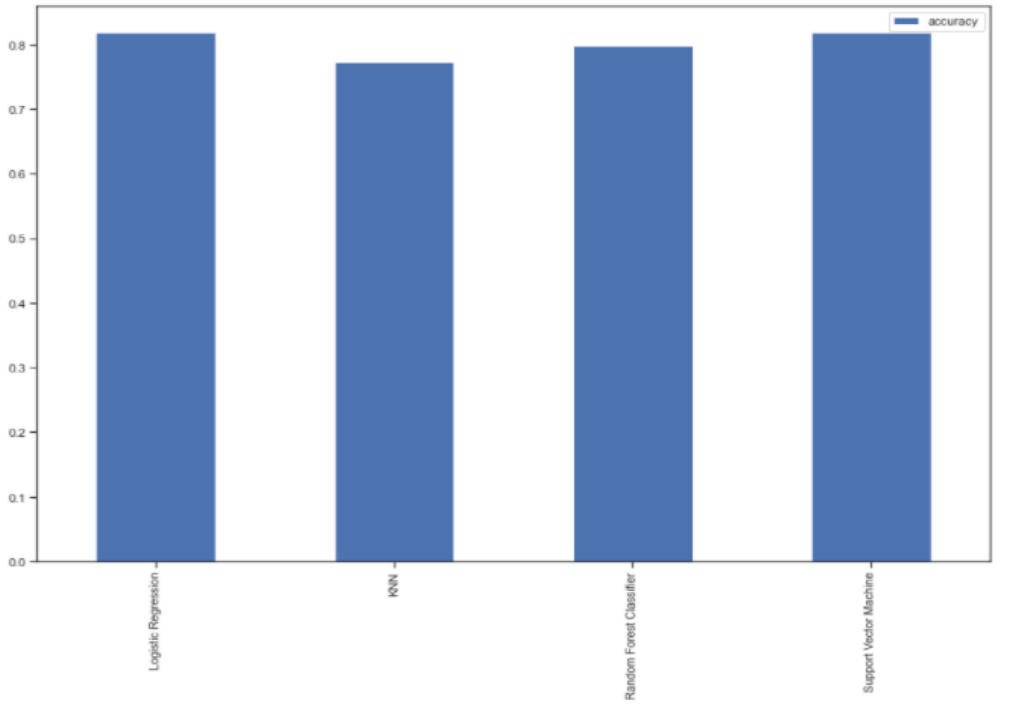
```
model_compare = pd.DataFrame({"Logistic Regression":log_reg,
                              "KNN":knn,
                              "Random Forest Classifier":clf,
                             "Support Vector Machine":svm,
                              },index=["accuracy"])
```

Comparison of my Machine Learning Models

[26]:

	Logistic Regression	KNN	Random Forest Classifier	Support Vector Machine
accuracy	0.818182	0.772727	0.798701	0.818182

[27]:
```
model_compare.T.plot.bar(figsize=(15,10));
```



Accuracy Scores

Classifier	Accuracy
Logistic Regression	81 %
KNN	77 %
Random Forest	79 %
Support vector machine (SVM)	81 %

HyperParameter Tunning using RandomizedSearchcv


```
# Tune LogisticRegression

np.random.seed(42)

# Setup random hyperparameter search for LogisticRegression
rs_log_reg = RandomizedSearchCV(LogisticRegression(),
                                param_distributions=log_reg_grid,
                                cv=5,
                                n_iter=20,
                                verbose=True)

# Fit random hyperparameter search model for LogisticRegression
rs_log_reg.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits

RandomizedSearchCV(cv=5, estimator=LogisticRegression(), n_iter=20,
                   param_distributions={'C': array([1.00000000e-04, 2.63665090e-04, 6.95192796e-04, 1.83298071e-03,
       4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790e-02,
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e+04]),
                                        'solver': ['liblinear']},
                   verbose=True)
```

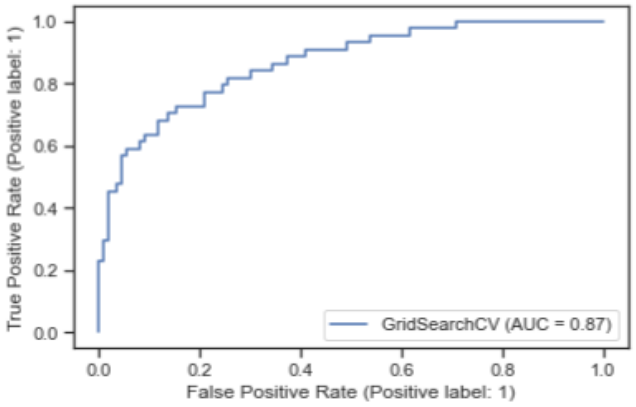
HyperParameter Tunning using Grid Search CV Logistic Regression

Category	Classifier	Pram-Distribution	CV	n-iter	verbose	5-Fold	Scores
Hyperparameter Tuning	Logistic Regression	Logistic_Reg Grid	5	20	TRUE	For each 20 Candidates totaling 100 fits	**0.831166**
Grid Search CV	Logistic regression	GridSearch_Log regression	5		TRUE	For each 30 candidates totaling 150 fits	**0.837662**

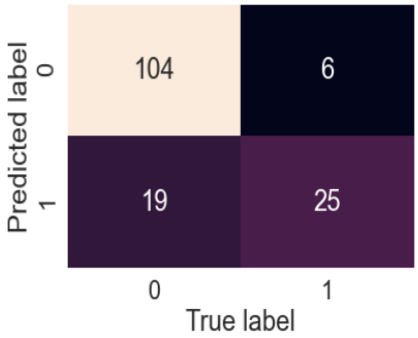
Best Model is Logistic Regression with 83% Accuracy Scores

ROC Curve

```
[35]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x18a2f2b74f0>
```



Confusion Matrix



Classification Report

	Precision	Recall	F1-score	Support
0	0.85	0.95	0.89	110
1	0.81	0.57	0.67	44
Accuracy			0.84	154
Macro Avg	0.83	0.76	0.78	154
Weighted avg	0.83	0.84	0.83	154

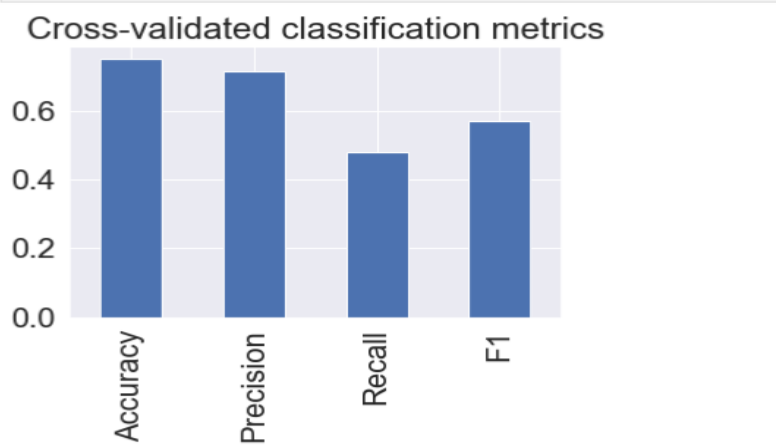
BEST PARAMETERS

{'C': 4.893900918477489, 'solver': 'liblinear'}

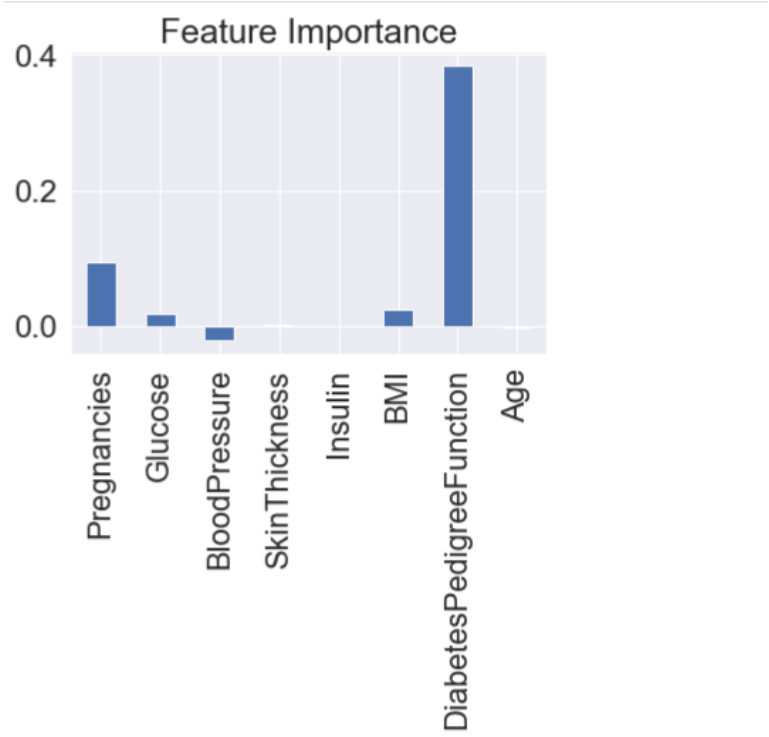
Chapter 3: Cross Validation Classification Scores

Cross Validation	CV Precision	CV Recall	CV F1-score
0.749743	0.7142036	0.5809116	0.569372

Visualize the CV Scores

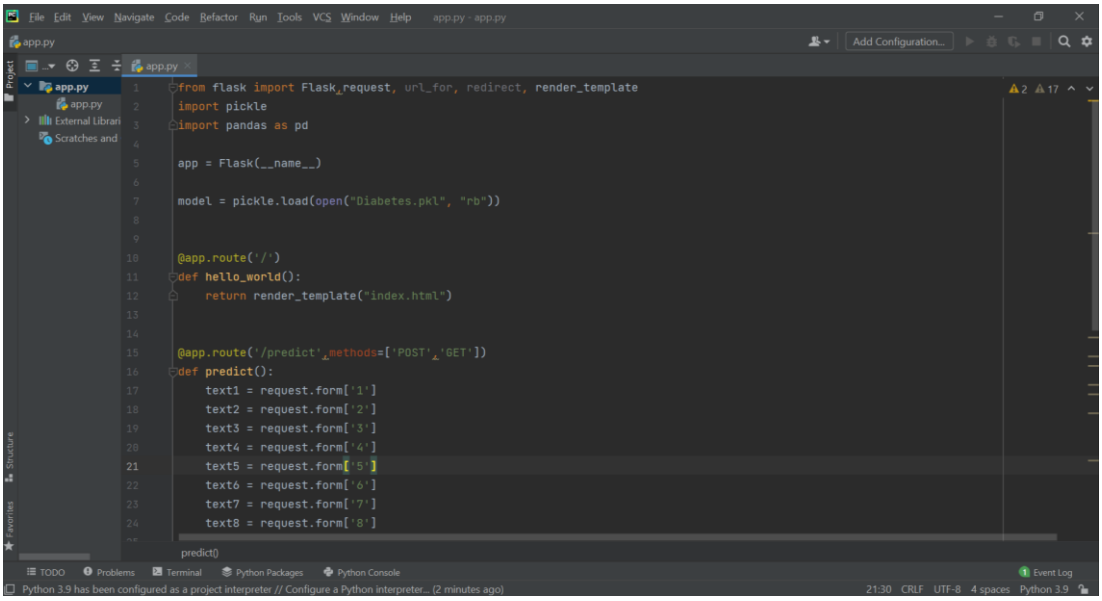


Feature Importance

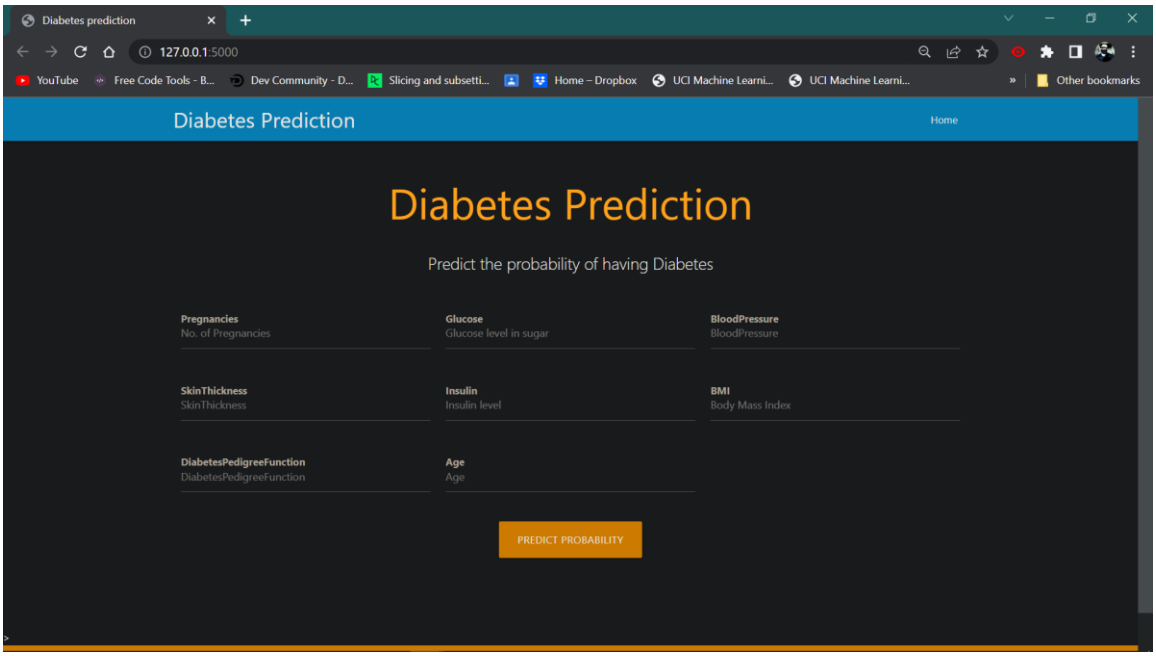


Chapter 4: **UI Diabetes Prediction**

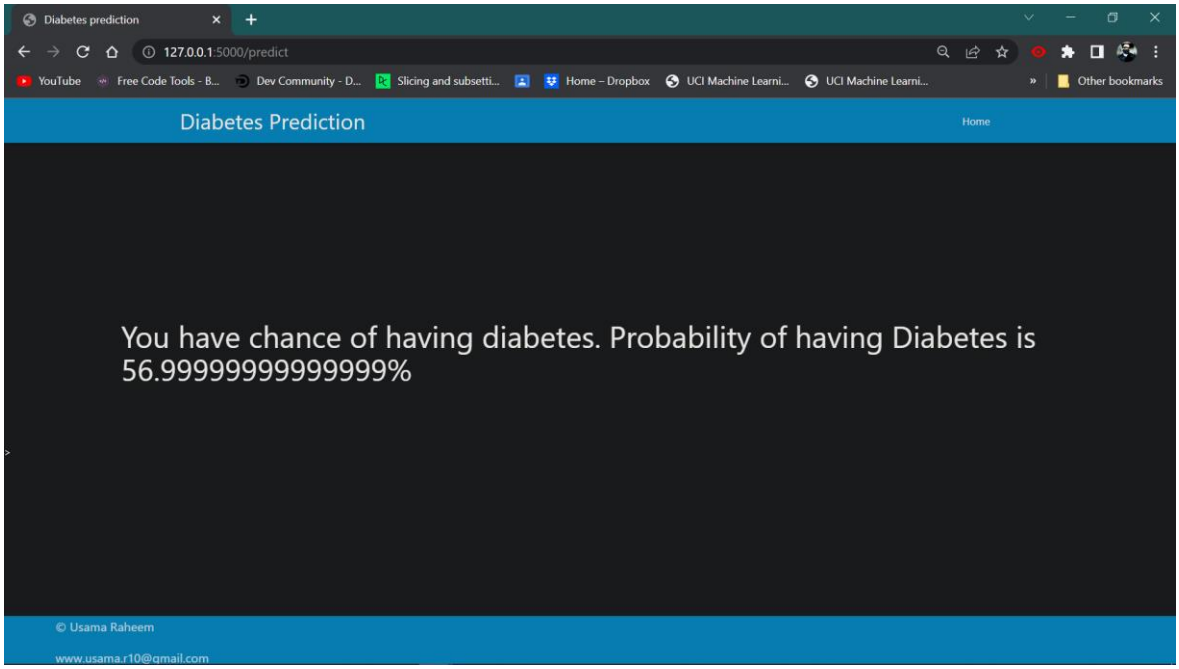
Deployment using Flask Framework after Saving Model with Pickle



Final UI of Diabetes Prediction using Machine Learning



Successful Prediction of Diabetes with %



Conclusion

Doing this Project is a fun activity, I learned so many new Machine learning and Data Science Techniques and Methods. After implementing all those concepts which were taught by my supervisor **Dr. Naveed Anwar Butt**, I am now Able to:

- Pre-process any Data file
- Perform Exploratory data Analysis on any given Data set
- Train and Test Models using various Machine learning Algorithms and Classifiers
- Generate Best Parameters using Hyper Parameter Tunning techniques
- Produce valuable outcomes from a Raw Data file for better decisions
- Deploy trained Model on Web With beautiful UI using Flask framework