



# DSCI 551 Final Report

Early Stage Diabetes Risk Prediction

-Praveen Iyer (USC ID: 5549510111; email: praveeni@usc.edu)

-Tejas Sujit Bharambe (USC ID: 2160849114; email: tbharamb@usc.edu)



---

# Title, Topic and Motivation



# Project Title

Early Stage Diabetes Risk Prediction

## Topic

Healthcare

## Dataset

[https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset.](https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset)



---

# Project Overview



# Motivation

1

Diabetes is a chronic (long lasting) health condition that affects how your body turns food into energy. As there isn't a cure yet for diabetes, diagnosing it at an early stage is essential to be clinically treated.

2

Due to the presence of a relatively long asymptomatic phase, early detection is a difficult task and hence we feel that if data science is used for this purpose it would be of immense social value.

3

Since diabetes is linked to the lifestyle of a person, based on a simple questionnaire we can predict whether a person is diabetic or not. This questionnaire includes critical questions like what is the age and gender of the person, whether the person is experiencing excessive urination or not, or is he or she experiencing excessive thirst or not and so on which help us predict whether the person is diabetic or not.



# Project Overview

**Data:** For this project we will need a labelled dataset which is open sourced. While looking to satisfy these requirements we found a dataset where data was collected using direct questionnaires and diagnosis results from the patients in the Sylhet Diabetes Hospital in Sylhet, Bangladesh. The dataset contains the signs and symptoms of newly diabetic or would be a diabetic patient.

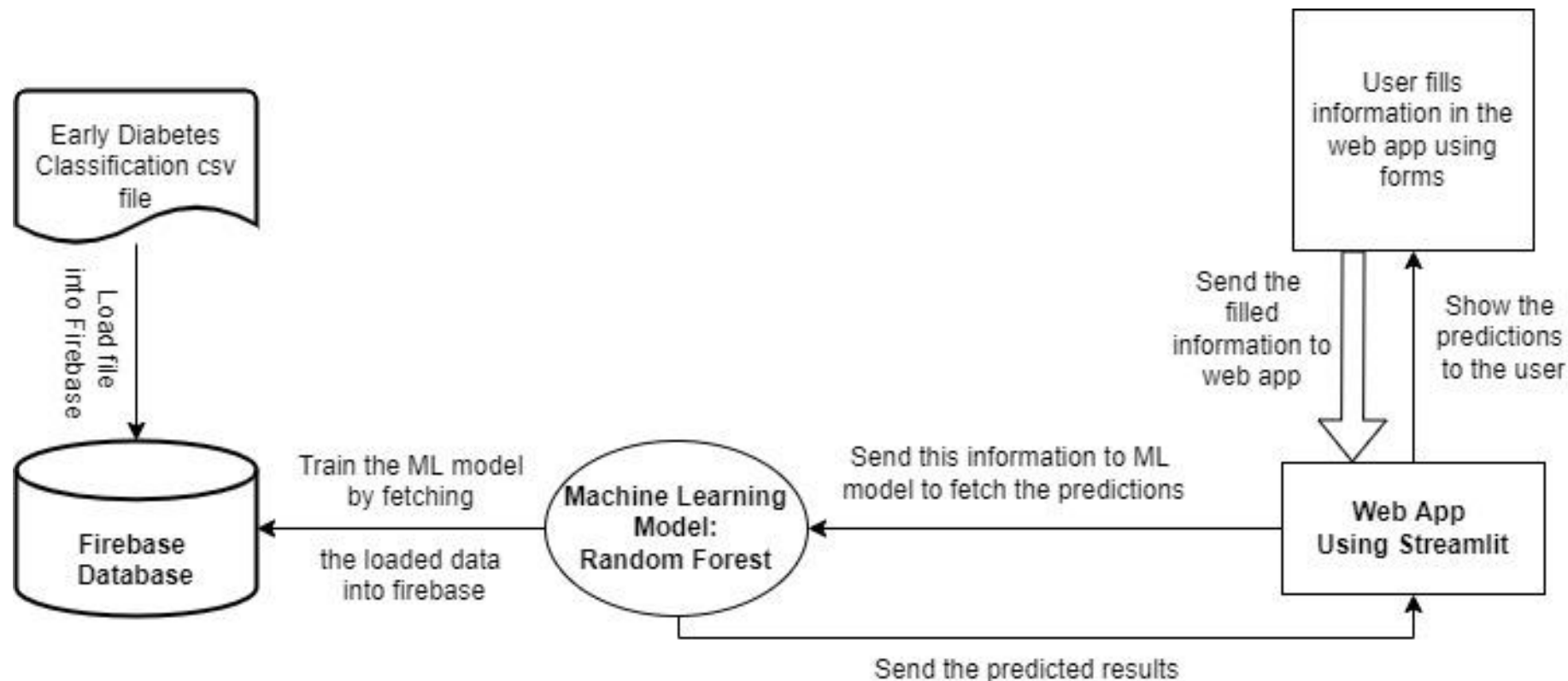
**Machine Learning Model:** We plan on using this publicly available dataset which has a list of carefully chosen 16 questions along with the corresponding diagnosis results to fit a carefully chosen and designed supervised machine learning classification algorithm like the random forest, naive bayes, neural networks and logistic regression . We are going forward with random forest as it gives the best accuracy scores.

**Web App (Browser):** To complement this we have developed a web based User Interface (UI) using streamlit where the users will be able to answer those 16 questions and see what our model predicts corresponding to their inputs (whether diabetic or not). Apart from this, the UI also has insightful data visualizations where we explore the most common features associated with diabetic risk.



# Data Flow

# Data Flow





---

# Project Architecture and Components along with screenshots



## Project Definition and Planning

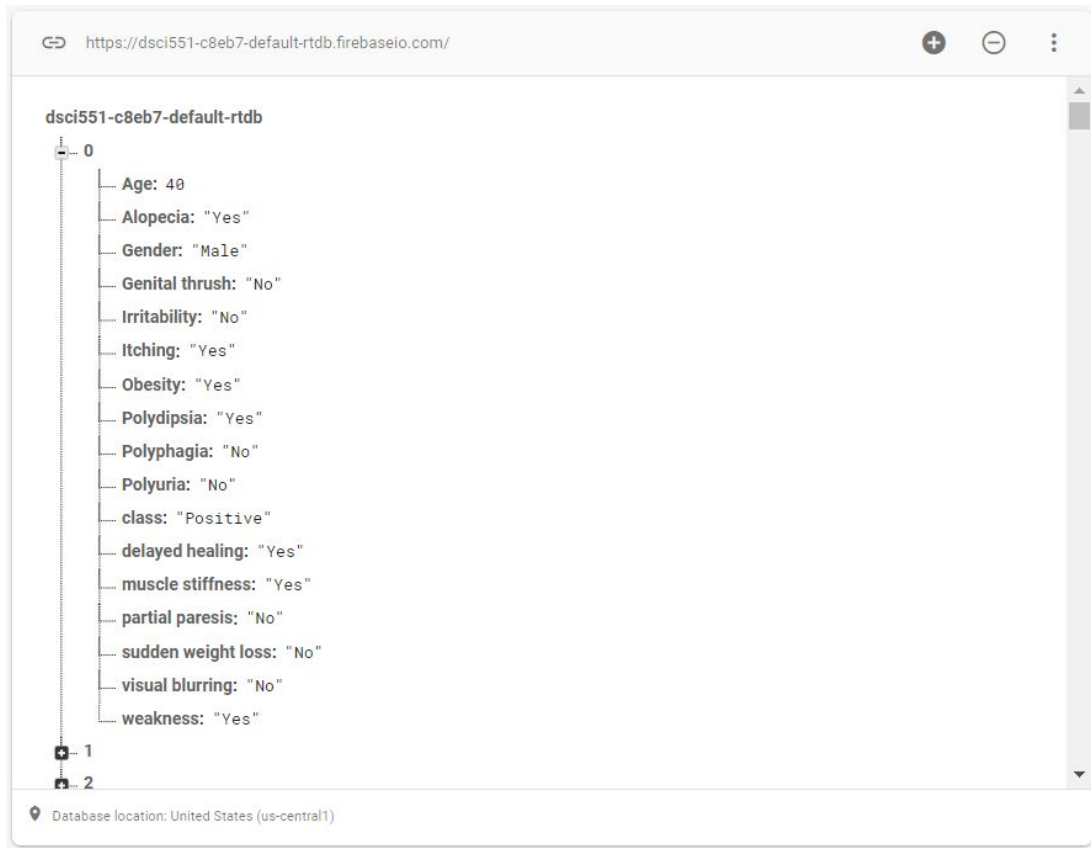
- Got a relatively clean dataset without missing values. (Using the link submitted in the project proposal)
- Explored mongoDB and firebase and used firebase as it is better suitable for small-scale applications like ours.
- Accessing data from firebase we experimented with multiple machine learning models ranging from logistic regression to neural networks. Based on the tradeoff between metrics (F1 score & accuracy) and interpretability we choose random forest model.
- Explored flask and streamlit and found streamlit to be more suitable for our purpose as we did not want to build highly customized interfaces and wanted a basic interface. Additionally streamlit also provides a pythonic way of implementation which we are most comfortable with.

# Firebase Database



## Project Execution (Cloud Database):

- Set up the Firebase successfully without any hassles
- Loaded the data into Firebase



# Snippets of loading and fetching data from firebase

Inserting data  
into firebase:

```
1 import pandas as pd
2 import json
3 import requests
4
5 path='diabetes_data_upload'
6 baseUrl='https://dsCi551-c8eb7-default-rtdb.firebaseio.com/'
7
8 def read_csv(path):
9     print('Reading csv data and converting into a dataframe')
10    return pd.read_csv(path+'.csv')
11
12 def df_to_json(dataframe):
13    print('Converting dataframe into json format')
14    output_json=dataframe.to_json(orient='index')
15    return json.loads(output_json)
16
17 def clear_database(baseUrl):
18    print('Deleting pre-existing records from the database')
19    response=requests.delete(url=baseUrl+'.json')
20
21 def append_data(baseUrl,json_data):
22    print('Uploading the json format into Firebase')
23    response=requests.put(url=baseUrl+'.json/',json=json_data)
24    print('Uploaded')
25
26 if __name__ == "__main__":
27     diabetes_data=read_csv(path)
28     diabetes_data_json=df_to_json(diabetes_data)
29     clear_database(baseUrl)
30     append_data(baseUrl,diabetes_data_json)
```

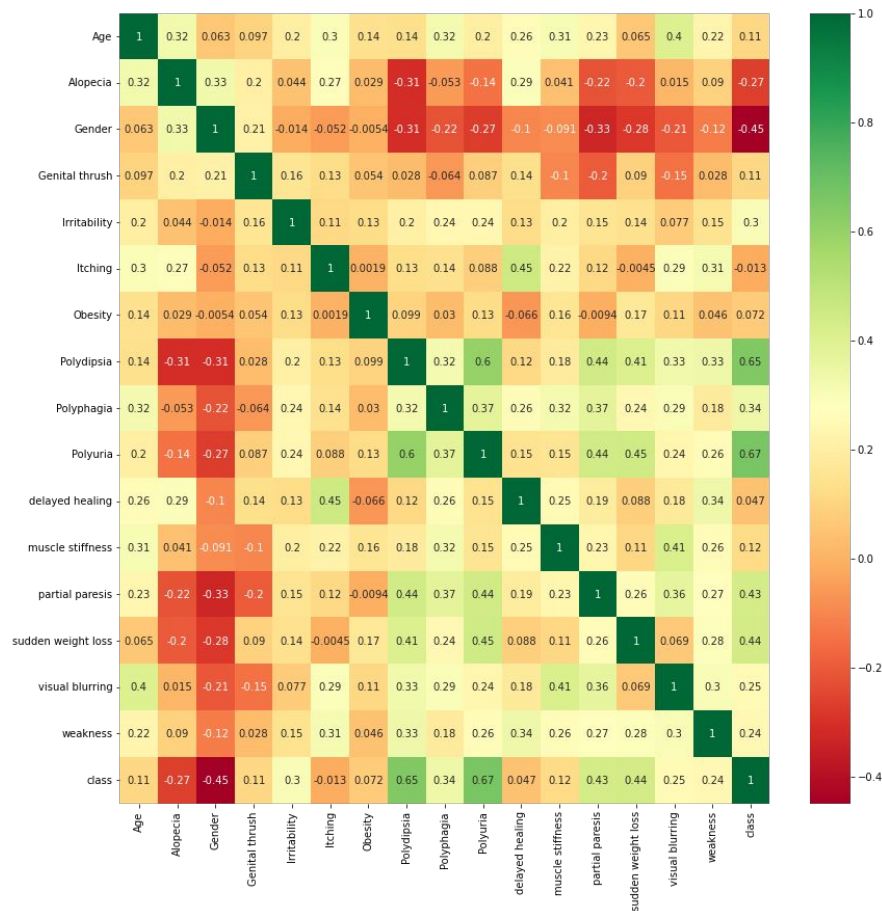
Fetching data  
from firebase:

```
19
20 firebase_url = 'https://dsCi551-c8eb7-default-rtdb.firebaseio.com/.json'
21 r = requests.get(firebase_url)
22 json_ob = json.loads(r.text)
23 df = pd.DataFrame.from_dict(json_ob)
24 df.replace(["Yes","Positive","Male"],1,inplace=True)
25 df.replace(["No","Negative","Female"],0,inplace=True)
26
```

# Exploratory Data Analysis



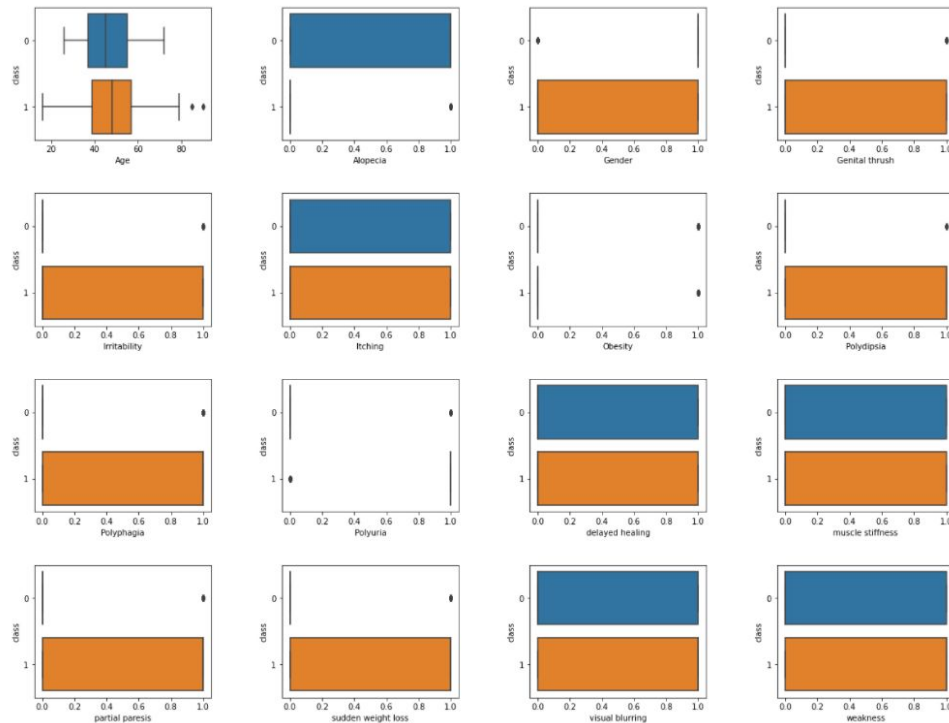
Plot for correlation between the features:



# Exploratory Data Analysis

## Exploratory Data Analysis

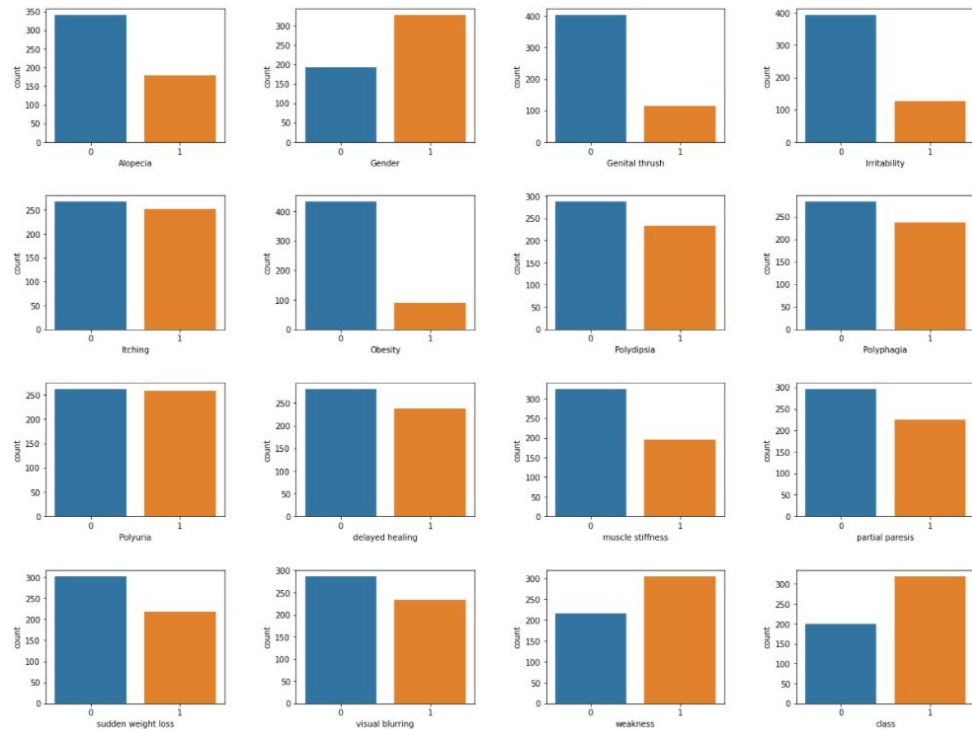
Box plot for all the features to check the distribution of data:



# Exploratory Data Analysis




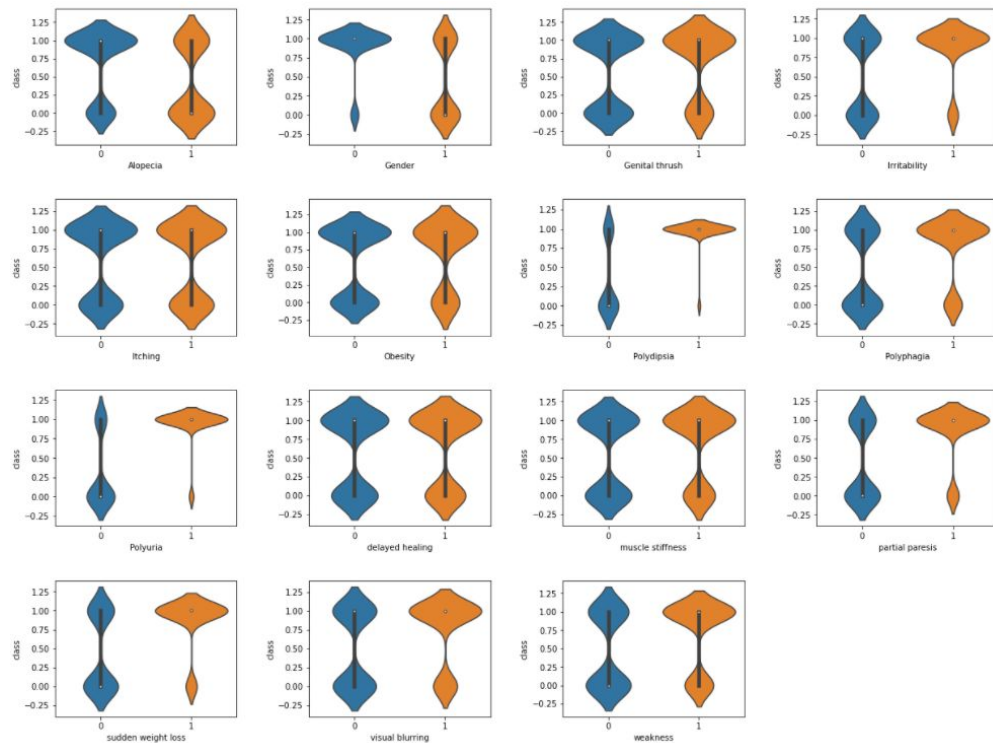
As almost all the features are categorical in nature, we plotted count plot for all the features:



# Exploratory Data Analysis



  
**Violin plot for all the features to plot the combination of box plot and kernel density estimate:**







## Project Execution (ML Models)

- Assigned values to categorical features and split the data into train and test samples to build and configure the model. We also checked for outliers in the dataset by plotting a boxplot and found out that there were not many outliers and we need not remove any samples.
- We plotted the correlation between all features and found out that there were not any features which were highly correlated and hence looking at the heatmap we could conclude that the features are more or less independent.
- Plotted the value counts for all categorical variables to check if any of them are imbalanced or not. (1:10 was the imbalance ratio we had in mind). Since all variables were balanced we proceeded to use them normally.
- To gauge the feature importance for our data we plotted the barplot and violin plot. In the bar plot we plotted the average value of each feature for a given value of class. If the feature is to be more important according to this plot then we expect to see stark differences in the averages for both class 0 and class 1. In the violin plot along with the box plot, we plotted the kernel density estimate. If the feature is to be more important according to this plot then we expect to see stark differences in the density distribution for each feature belonging to class 0 and class 1. We found out that features like polydipsia and polyuria are the best features followed by sudden weight loss and partial paresis.

# Implementation of Machine Learning Models



- 
- A decorative horizontal bar with a teal segment on the left and an orange segment on the right.
- *Trained multiple machine learning models to understand which model fits the best to our data.*

*The details of the models we tried are listed below.*

- **Random Forest** - We felt that random forest being a tree based method might perform well on these dataset as we have a lot of categorical variables and hence a tree based based method would work well. Based on the cross validation scores, we used 120 decision trees and fixed the maximum depth of each tree to 3 for our random forest model.
- **Logistic regression** - Initially, we used the default scikit parameters for this logistic regression model ('l2' penalty with 'lbfgs' solver). This gave reasonably good results but we still saw scope for improvement and hence went on to try other models.

# Implementation of Machine Learning Models



- **Categorical Naives Bayes** - As our features were more or less independent, we thought of implementing Naive Bayes. Since all the features are categorical in our case, we have used the categorical naive bayes algorithm as it assumes that each feature has its own categorical distribution.
- **Neural network** - To experiment with deep learning based tasks for this task, we built a simple network with the following architecture:

```
my_network(  
    (l1): Linear(in_features=16, out_features=128, bias=True)  
    (a): ReLU()  
    (l2): Linear(in_features=128, out_features=32, bias=True)  
    (l3): Linear(in_features=32, out_features=1, bias=True)  
    (a_final): Sigmoid()  
    (d): Dropout(p=0.3, inplace=False))
```

We used a high dropout probability of 0.3 to make sure that our network does not overfit the data. We used binary cross entropy loss along with the Adam optimizer with a learning rate of 0.005 to train the network. We trained the model for 20 epochs and used a batch size of 32 to pass our data to the model which was built using the well renowned deep learning framework PyTorch.

# Comparison of results from ML models



Model Comparison	F1 Score		Accuracy	
ML Models	Train	Test	Train	Test
Random Forest	0.932	0.979	0.916	0.974
Logistic Regression	0.943	0.969	0.930	0.962
Categorical Naives Bayes	0.901	0.979	0.885	0.974
Neural Network	0.927	0.989	0.914	0.987

***Neural network outperforms other algorithms when it comes to test accuracy and F1 score but we decided to go ahead with random forest due to its superior interpretability without much loss in metrics.***

# Snippets of the Random Forest model used



## Model Building:


```
30
31 all_x_columns = df.columns.values.tolist()
32 all_x_columns.remove("class")
33 x = df.loc[:,all_x_columns]
34 y = df.loc[:,"class"]
35
36 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=rs, stratify=y)
37
38 def get_metrics_for_model(model):
39     model.fit(x_train,y_train)
40     train_pred = model.predict(x_train)
41     test_pred = model.predict(x_test)
42     train_f1 = f1_score(y_train, train_pred, average='binary')
43     test_f1 = f1_score(y_test, test_pred, average='binary')
44     train_acc = accuracy_score(y_train, train_pred)
45     test_acc = accuracy_score(y_test, test_pred)
46     return train_f1,test_f1,train_acc,test_acc
47
48 model_rf = RandomForestClassifier(n_estimators=120, max_depth=3, oob_score=False, n_jobs=-1, random_state=rs)
49 train_f1_rf,test_f1_rf,train_acc_rf,test_acc_rf = get_metrics_for_model(model_rf)
50 print(f"\ntrain_f1_rf {train_f1_rf} , test_f1_rf {test_f1_rf}")
51 print(f"train_acc_rf {train_acc_rf} , test_acc_rf {test_acc_rf}")
```

## Model Prediction:

```
195 def get_feature_importances():
196     importances = model_rf.feature_importances_
197     std = np.std([tree.feature_importances_ for tree in model_rf.estimators_], axis=0)
198     return importances,std,all_x_columns
199
200 def get_prediction(x):
201     # y = model_nn(torch.from_numpy(x).float())
202     y = model_rf.predict(x.reshape(1,-1))
203     # if y.item()>=0.5:
204     if y>=0.5:
205         return "Positive"
206     else:
207         return "Negative"
```

# Implementation of Web App



- 
- A horizontal bar with a teal segment on the left and an orange segment on the right.
- To display our full web based UI with all functionalities, we have to run the following command in the terminal (in the directory having all our files): `streamlit run pass2_main.py`
  - This calls our main function which has the MutiApp class which in turn adds all the content of all 3 pages based on the radio button selection.
  - When the Self diagnosis page is called, our pass1.py file (which has our ML model and firebase data access code available) is invoked and we can get the relevant predictions based on user inputs (the user can also see the help information available with each button to better understand the questionnaire).
  - Additionally the introduction page content and the Early diabetes page content is also loaded and displayed whenever the appropriate radio button is selected.
    -

# Snippets of the Web App Components



**Multi Page Component:** As our web app has multiple pages, we had to configure it accordingly

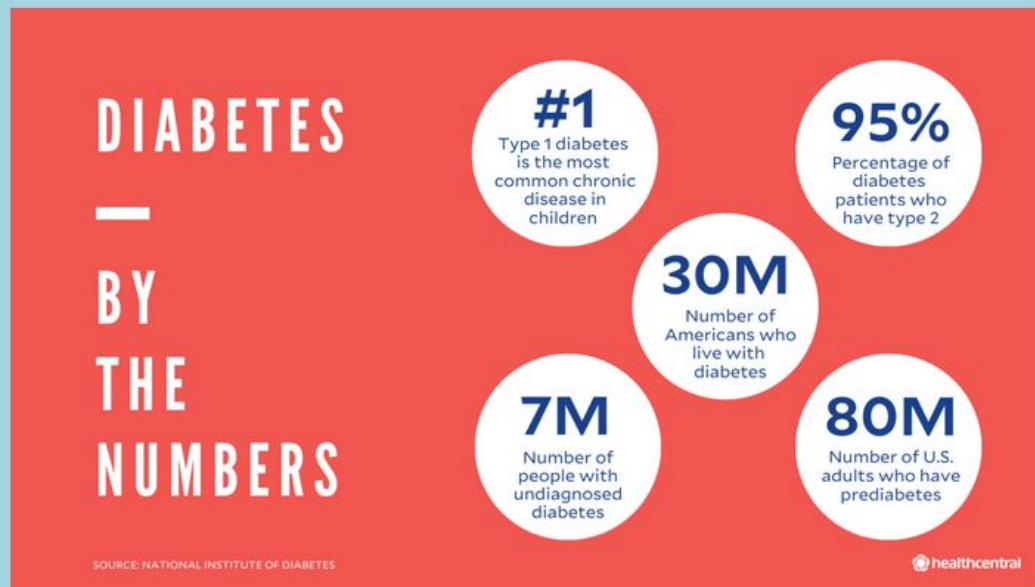
```
1 from pass2_intro import intro_content
2 from pass2_model_form_and_prediction import display_cells_to_get_data
3 from pass2_early_diabetes import get_early_diabetes_page
4 import streamlit as st
5
6 class MultiApp:
7     def __init__(self):
8         self.apps = []
9
10    def add_app(self, title, func):
11        self.apps.append({
12            "title": title,
13            "function": func
14        })
15
16    def run(self):
17        st.write('<style>div.row-widget.stRadio > div{flex-direction:row;justify-content: center;} </style>', unsafe_allow_html=True)
18        st.write('<style>div.st-bf{flex-direction:column;} div.st-ag{font-weight:bold;padding-left:2px;}</style>', unsafe_allow_html=True)
19        choose=st.radio("",self.apps,format_func=lambda app: app['title'])
20        choose['function']()
21
22 app = MultiApp()
23 app.add_app(" Introduction", intro_content)
24 app.add_app("Early Diabetes", get_early_diabetes_page)
25 app.add_app("Self diagnosis", display_cells_to_get_data)
26 # The main app
27 app.run()
```

# Snippets of the Web App (Page 1 - Introduction)



☒ Introduction ☐ Early Diabetes ☐ Self diagnosis

Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy. Most of the food you eat is broken down into sugar (also called glucose) and released into your bloodstream. When your blood sugar goes up, it signals your pancreas to release insulin. Insulin acts like a key to let the blood sugar into your body's cells for use as energy.



Statistics on diabetes



# Snippets of the Web App Code (Page 1 - Introduction) ≡

Introduction  
Page:

```
1 import streamlit as st
2 from PIL import Image
3
4 def intro_content():
5     with open("diabetes_intro_writeup.txt") as f:
6         text = f.read().split("\n")
7         st.write(text[0])
8         image = Image.open('diabetes_stats.png')
9         st.image(image, caption='Statistics on diabetes')
```

Introduction Page Write Up:

```
1 Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy. Most of the food you eat is broken
down into sugar (also called glucose) and released into your bloodstream. When your blood sugar goes up, it signals your pancreas to release
insulin. Insulin acts like a key to let the blood sugar into your body's cells for use as energy.
```

# Snippets of the Web App Code (Page 2 - Early Diabetes)

☐ Introduction ☒ Early Diabetes ☐ Self diagnosis

When diabetes is detected early, the patient may not only delay but even prevent progression to diabetes. Disease prevention is significantly less expensive and a lot of future diabetic complications can be avoided. Hence both monetarily and health wise, from both the patients point of view and the healthcare system's point of view early diabetes detection is a very key challenge having significant impacts.

## Potential Warning Signs of Diabetes

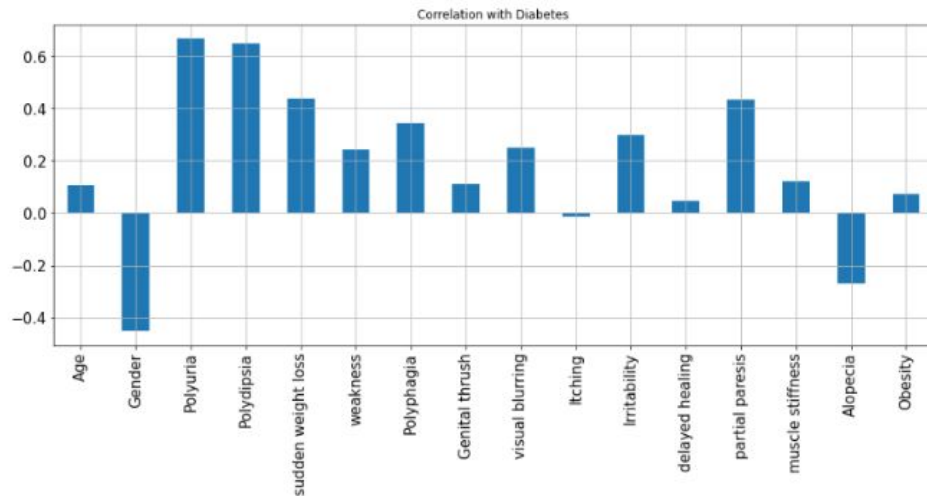


## DIABET SYMPTOMS

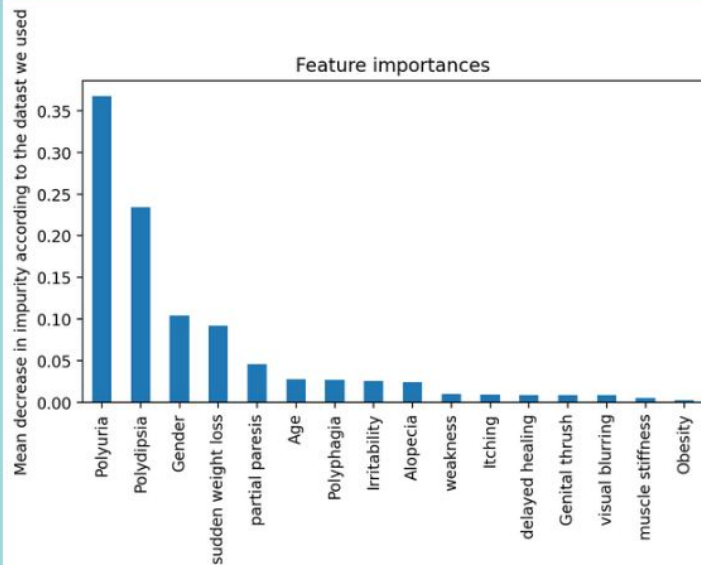


Symptoms of Early diabetes

Diabetes is one of the fastest growing chronic life threatening diseases that has already affected millions of people worldwide. Due to the presence of a relatively long asymptomatic phase, early detection of diabetes is always desired for a clinically meaningful outcome. Around 50% of all people suffering from diabetes are undiagnosed because of its long-term asymptomatic phase. To tackle this problem we used a publically available dataset which contains 520 observations with 17 characteristics, collected using direct questionnaires and diagnosis results from the patients in the Sylhet Diabetes Hospital in Sylhet, Bangladesh. Based on this dataset, we tried to plot the correlation and feature importances as depicted in the figures below.



# Snippets of the Web App Code (Page 2 - Early Diabetes)



The glossary has been listed below to help you understand the plot better.

- Polyuria: Whether the patient experienced excessive urination or not.
- Polydipsia: Whether the patient experienced excessive thirst/excess drinking or not.
- Sudden\_weight\_loss: Whether patient had an episode of sudden weight loss or not.
- Partial paresis: Whether patient had partial or mild paralysis
- Polyphagia: Whether patient had an episode of excessive/extreme hunger or not.
- Alopecia: Whether patient had an episode of hair loss.
- Genital\_thrush: Whether patient had a yeast infection or not.
- Visual\_blurring: Whether patient had an episode of blurred vision.

To add social value we have decided to build a free to use tool in which you have to answer a few questions based on which you will get your diagnosis. Please note that this diagnosis is an indicative one and cannot be completely relied upon. Kindly visit a medical expert for more comprehensive insights.

# Snippets of the Web App Code (Page 2 - Early Diabetes)

## Early Diabetes Page:

```
1 import streamlit as st
2 from pass1 import get_feature_importances
3 from PIL import Image
4 import pandas as pd
5 import matplotlib.pyplot as plt
6
7 def get_early_diabetes_page():
8     with open("early_diabetes writeup.txt") as f:
9         text = f.read().split("\n")
10        st.write(text[0])
11        early_image = Image.open('early_diabetes.jpg')
12        st.image(early_image)
13        prevention_image = Image.open("diabetes_symptoms.jpg")
14        st.image(prevention_image, caption='Symptoms of Early diabtetes')
15
16        st.write(text[1])
17
18        corr_im = Image.open("Correlation_Plot_Diabetes.PNG")
19        st.image(corr_im)
20
21        importances,std,feats = get_feature_importances()
22        forest_importances = pd.Series(importances, index=feats)
23        forest_importances.sort_values(ascending=False,inplace=True)
24        fig, ax = plt.subplots()
25        # forest_importances.plot.bar(yerr=std, ax=ax)
26        forest_importances.plot.bar(ax=ax)
27        ax.set_title("Feature importances")
28        ax.set_ylabel("Mean decrease in impurity according to the datast we used")
29        fig.tight_layout()
30        st.pyplot(fig)
31        st.write(text[2])
32        for i in range(8):
33            st.write(text[3+i])
34        st.write(text[-1])
```

# Snippets of the Web App Code (Page 3 - Self Diagnosis )

☐ Introduction ☐ Early Diabetes ☒ Self diagnosis

## Early diabetes prediction

Enter your age here (as an integer).



Please select your gender?

Male



Do you urinate excessively?



Yes



Do you feel abnormally thirsty?



Yes



Have you experienced sudden weight loss recently?



Yes



Have you recently experienced any hair loss?



Yes



Are you obese?

No



[Click here to get your diagnosis](#)

Your diagnosis is Negative

# Snippets of the Web App Code (Page 3 - Self Diagnosis )

Are you obese?

Yes

[Click here to get your diagnosis](#)

Your diagnosis is Positive

## TREATMENT AND PREVENTION



MEDICATIONS



VISIT A DOCTOR



DIAGNOSTIC



EXERCISE



HEALTHY DIET



NO SMOTING



INSULIN  
INJECTIONS



AVOID ALCOHOL



FOOD CONTROL



KEEP NORMAL  
WEIGHT



# Snippets of the Web App Code (Page 3 - Self Diagnosis)

## Self Diagnosis Page:

```
1 import numpy as np, pandas as pd, streamlit as st, matplotlib.pyplot as plt
2 from PIL import Image
3 from passl import get_prediction
4 def display_cells_to_get_data():
5     st.title("Early diabetes prediction")
6     with st.form("diabetes_symptoms"):
7         flag = True
8         age = st.text_input("Enter your age here (as an integer).",help="Pease enter an integer")
9         age = age.strip()
10        if len(age) > 0:
11            if not age.isnumeric():
12                flag = False
13                st.error("Age is not a number")
14            else:
15                age = int(age)
16        gender = st.selectbox('Please select your gender?',['Male', 'Female'])
17        polyuria = st.selectbox('Do you urinate excessively?',['Yes', 'No'],help="Whether the patient experienced excessive urination or not.")
18        polydipsia = st.selectbox('Do you feel abnormally thirsty?',['Yes', 'No'],help="Whether the patient experienced excessive thirst/excess drinking or not.")
19        sudden_weight_loss = st.selectbox('Have you experienced sudden weight loss recently?',['Yes', 'No'],help="Whether patient had an episode of sudden weight loss.")
20        weakness = st.selectbox('Do you feeling weak or fatigued often?',['Yes', 'No'])
21        polyphagia = st.selectbox('Do you eat excessively?',['Yes', 'No'],help="Whether patient had an episode of excessive/extreme hunger or not.")
22        genital_thrush = st.selectbox('Do you have any infections, especially near your genitals and/or mouth?',['Yes', 'No'],help="Whether patient had a yeast infection.")
23        visual_blurring = st.selectbox('Do you feel like your vision has blurred recently?',['Yes', 'No'],help="Whether patient had an episode of blurred vision.")
24        itching = st.selectbox('Do you face localized and severe itching anywhere on your body?',['Yes', 'No'])
25        irritability = st.selectbox('Do you feel like you have low mood and are irritable?',['Yes', 'No'])
26        delayed_healing = st.selectbox('Do you feel that your recent wounds have healed slowly or not healed at all?',['Yes', 'No'])
27        partial_paresis = st.selectbox('Have you recently experienced weakening of muscles or a group of muscles?',['Yes', 'No'],help="Whether patient had partial paralysis.")
28        muscle_stiffness = st.selectbox('Have you recently experienced cramps, joint pains or painful walking?',['Yes', 'No'])
29        alopecia = st.selectbox('Have you recently experienced any hair loss?',['Yes', 'No'],help="Whether patient had an episode of hair loss.")
30        obesity = st.selectbox('Are you obese?',['Yes', 'No'])
31        feats = ['age', 'alopecia', 'gender', 'genital_thrush', 'irritability', 'itching', 'obesity', 'polydipsia', 'polyphagia', 'polyuria', 'delayed_healing']
32        data = []
33        for feat in feats:
34            if eval(feat)=="Yes" or eval(feat)=="Male":
35                feat = 1
36            elif eval(feat)=="No" or eval(feat)=="Female":
37                feat = 0
38            exec(f"data.append({feat})")
39        if st.form_submit_button('Click here to get your diagnosis'):
40            if not flag or len(str(age))==0:
41                st.write("Please check your inputs again.")
42            else:
43                x = np.array(data)
44                result = get_prediction(x)
45                st.success('Your diagnosis is {}'.format(result))
46                if result=="Positive":
47                    prevention_im = Image.open("diabetes_prevention.jpg")
48                    st.image(prevention_im)
```

# Snippets of the Web App Configuration



App  
Configuration:

```
1  [theme]
2  base="light"
3  backgroundColor="#a0d6e0"
4  textColor="#000000"
```



# Appendix: Snippets of neural networks

Code for other models like neural network:

## Snippets of the code:

```
class my_network(nn.Module):
    def __init__(self, input_size, dropout_required=True):
        super(my_network, self).__init__()
        self.input_size = input_size
        self.dropout_required = dropout_required
        self.l1 = nn.Linear(input_size, 128)
        self.a = nn.ReLU()
        self.l2 = nn.Linear(128, 32)
        self.l3 = nn.Linear(32, 1)
        self.a_final = nn.Sigmoid()
        if dropout_required:
            self.d = nn.Dropout(p=0.3)

    def forward(self, x):
        out = self.l1(x)
        out = self.a(out)
        if self.dropout_required:
            out = self.d(out)
        out = self.l2(out)
        out = self.a(out)
        if self.dropout_required:
            out = self.d(out)
        out = self.l3(out)
        out = self.a_final(out)
        return out
```

```
fls, accs = [], []
def train_net(data_batches, loss_function, optim):
    n_samples = len(data_batches.dataset)
    model_nn.train()
    for batch, (x, y) in enumerate(data_batches):
        x, y = x.to(device).float(), y.to(device).float()
        pred = model_nn(x)
        pred = pred.squeeze()
        loss = loss_function(pred, y)

        optim.zero_grad()
        loss.backward()
        optim.step()

        if batch % 5 == 0:
            loss, done = loss.item(), batch * len(x)
            print(f"Loss={loss} for progress {done}/{n_samples}")

def test_net(data_batches, loss_function):
    n_samples = len(data_batches.dataset)
    n_batches = len(data_batches)
    model_nn.eval()
    test_loss, correct = 0, 0
    tp, tn, fp, fn = 0, 0, 0, 0
    with torch.no_grad():
        for x, y in data_batches:
            x, y = x.to(device).float(), y.to(device).float()
            pred = model_nn(x)
            pred = pred.squeeze()
            test_loss += loss_function(pred, y)
            pred = (pred >= 0.5).int()
            pred, y = pred.cpu().detach().numpy(), y.cpu().detach().numpy()
            cm = confusion_matrix(y, pred)
            tn += cm[0][0]
            fn += cm[1][0]
            tp += cm[1][1]
            fp += cm[0][1]
            correct += ((pred == y).sum()).item()
    test_loss = test_loss / n_batches
    accuracy = correct / n_samples
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
    f1 = (2 * precision * recall) / (precision + recall)
    fls.append(f1)
    accs.append(accuracy)
    print(f"Test error is {test_loss} ; Accuracy is {accuracy} ; F1 score is {f1}")
```

---

# Reflection on learning experience

# Reflection on learning experience



- We gained experiences in building web based UI which neither of us had any prior experiences in. This is an essential skill set which we feel can be leveraged at multiple points in the future. Getting started with complex functionalities of streamlit is also something that has been a steep learning curve for both of us.
- We also explored and learnt the basics of pytorch to implement our deep learning based models which is a widely used industry standard library in the field of deep learning.
- We got ourselves familiarized with firebase as all our data was stored in it. Storing and accessing all our data from the cloud was something that was a new experience and we enjoyed learning.
- Through this project, we learnt essential soft skills like debugging and exploring and working on new fields like Web based UI development.

---

# Challenges Faced

# Challenges Faced



- While trying to zero in on our Machine Learning model, we decided to initially choose the deep learning based neural network model because of its superior metric based performance on our test dataset. However, we later switched to the random forest model as we felt that slightly lower metric performance was made up by its much superior interpretability options. This we felt could be useful for users without much prior knowledge and provide them with valuable insights.
- While designing our Web based UI, we faced multiple challenges in streamlit. Since streamlit is not as customizable, to make our UI pleasing to eyes we wanted to choose an appropriate background color. However, to do this we were not able to find any functionality in streamlit which directly enabled us to do this and as streamlit is a relatively small scale project it did not provide sufficiently accessible documentation to help us tackle this problem. Eventually we figured out that we can change the config file corresponding to streamlit which enables us to set multiple attributes like the background color, secondary background color and text color.

# Challenges Faced



- To make multiple pages a part of our UI we were not getting anything initially. However as a workaround we wrote our own python class which makes use of radio buttons to implement a page each for each of the radio button options. This was a crucial step in our UI as we wanted multiple pages to be a part of our UI for our visualized idea.
- While building our MultiApp class we chose to use radio buttons which by default are oriented in vertical direction in streamlit. However it did not make sense from an aesthetic point of view to have the radio button vertically oriented. We searched hard for solutions to reorient the button horizontally but were not immediately able to find any. However, we eventually found a smart workaround by using some short HTML snippets to reorient the radio button horizontally and get what we desired.



---

# Team Members & Responsibilities



# Team

Praveen Iyer | MS Applied Data Science | Email ID - [praveeni@usc.edu](mailto:praveeni@usc.edu) | USC ID - 5549510111

Tejas Sujit Bharambe | MS Applied Data Science | Email ID - [tbharamb@usc.edu](mailto:tbharamb@usc.edu) | USC ID - 2160849114



## Project Plan, Milestones & Team Members' Responsibilities

[illegible]



## Relevant Papers

- Likelihood Prediction of Diabetes at Early Stage Using Data Mining Techniques
  - [\[Web Link\]](#) - Computer Vision and Machine Intelligence in Medical Image Analysis. Springer, Singapore, 2020. 113-125
  - Authors and affiliations:
    - M. M. Faniqul IslamEmail
    - Rahatara Ferdousi
    - Sadikur Rahman
    - Humayra Yasmin Bushra
- [https://link.springer.com/chapter/10.1007/978-981-16-5655-2\\_41](https://link.springer.com/chapter/10.1007/978-981-16-5655-2_41)



# Google Drive References to all relevant files - code, data, images and documents

Link - Click [here](https://drive.google.com/drive/folders/18-IHWW6w87gheICBjj1bes-Wj2gBWVM4?usp=sharing) to access the google drive

<https://drive.google.com/drive/folders/18-IHWW6w87gheICBjj1bes-Wj2gBWVM4?usp=sharing>



# Thank you.

