Report for Research Internship

By:

Rayhaan Khan working under Shadab Khan, SVP of AI at M42 – Abu Dhabi

At:

Abu Dhabi

Signature:

Introduction:

Today, it seems as if the word "Diabetes" is almost everywhere, from doctors to pharmaceutical advertisements. But have you ever wondered what it is really? Diabetes, as described by the World Health Organization (WHO), Is a condition, where the pancreas in our body fails to produce sufficient quantities of insulin, the hormone which is responsible for the breakdown of glucose (sugars a person consumes):

Before we proceed further, we need to know that there are two types of diabetes, Type-1 and Type-2. Let's dig a bit deeper into them.

Type-1:

This kind of diabetes is usually regarded as a 'genetic disease' meaning that it is passed on to the offspring from its parents. These patients need to be injected with insulin every day to prevent too much glucose concentration in their blood, as this can damage the nerves of the body.

Type-2:

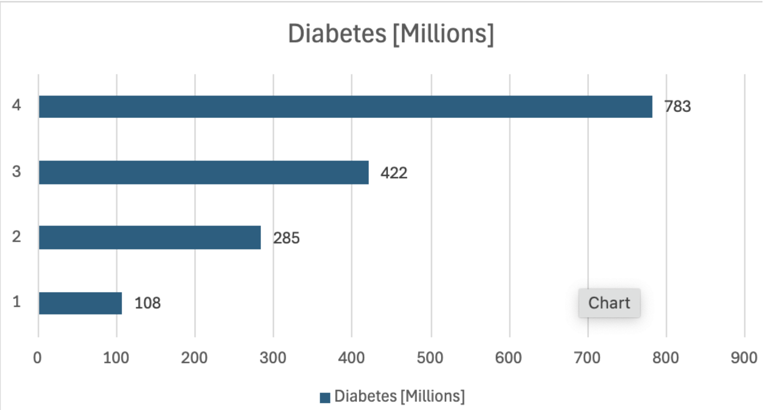
This type is the most common. According to WHO, 95% of the people suffering from diabetes are type-2 diabetics. In this, the body is not able to properly utilize the insulin produced by the pancreas. Unlike type-1 diabetes, this is preventable, having sufficient exercise and a healthy and balanced diet are the best ways to prevent type-2 diabetes.

Some symptoms of diabetes can include but are not limited to:

- feeling very thirsty
- needing to urinate more often than usual
- blurred vision
- feeling tired
- losing weight unintentionally

Over time, diabetes can damage blood vessels in the heart, eyes, kidneys, and nerves. People with diabetes have a higher risk of health problems including heart attack, starstruck, and kidney failure. Diabetes can cause permanent vision loss by damaging blood vessels in the eyes. In some severe cases, people with diabetes develop problems with their feet from nerve damage and poor blood flow. This can cause foot ulcers and may lead to amputation.

Around the world, according to the last 2019 report by WHO, it is estimated that 422 million individuals are suffering from diabetes. 21.1 million of them suffer from type-1 diabetes and the rest suffer from type-2 diabetes. Through recent studies, it is estimated that the number of people suffering from diabetes is projected to grow to 783 million in 2045.



[Legend, 1: 1980 | 2: 2014 | 3: 2019| 4: 2045]

By looking at the graph above, it is clear that the number of people with diabetes is increasing. And it is a worrying trend, because it is estimated that each year around 1.5 million people (about the population of Nebraska) die directly due to diabetes. [0]

This increase can be largely attributed to people nowadays living a more sedentary lifestyle, and unhealthy eating habits which include excessive sugar consumption, and sodium consumption. [1]

But, before someone becomes a diabetic, they pass through it a phase called the **pre-diabetes stage** which as the name suggests occurs just before type-2 diabetes. In this phase, an individual can come out of the diabetic spectrum before they become diabetic. By having a healthy lifestyle, getting frequent exercise, and sleeping well.

Why am I telling you this?

As you can imagine, this is a serious issue that needs to be solved before someone becomes a diabetic. So, to identify people who are on the diabetic spectrum early, I have created an ML (Machine Learning) model from scratch, using publicly available data from the National Health And Nutrition Examination Survey, (Referred to as NHANES). This model gives the probability of someone being on the diabetic spectrum by taking values, which can be input by the doctor or some individual if the individual gets a probability of 1, it means that they are a diabetic already. If they get above 0.5, it means that they are in the pre-diabetic range and should start healthy habits. Also, I want to use this opportunity to showcase how well models can be trained and can perform using publicly available data.

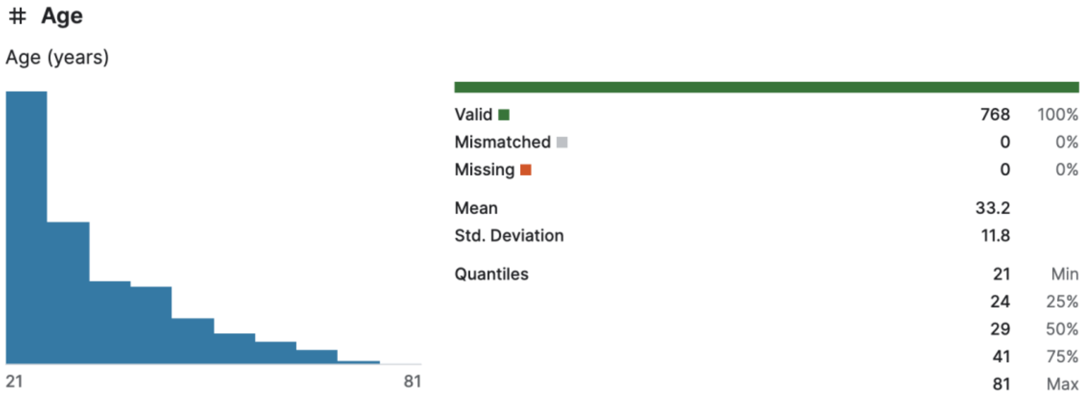
Methods used for training the model

The most important part when training any model is the data which the model is being trained upon. So naturally, having the right data is key. For this, I chose the NHANES dataset from Kaggle, in addition to being very high quality, as mentioned above it is also publicly available and is free to use by everyone.

Here are some demographics: -

Age

As you can see, most people reviewed are in the range of between 21 – 33 years. The overall range is between 21-81 years. However, the mode is 21 years old. Age is an important factor in determining if a patient has diabetes or not because as the body becomes old, the amount of insulin secreted by the pancreas gets lower.



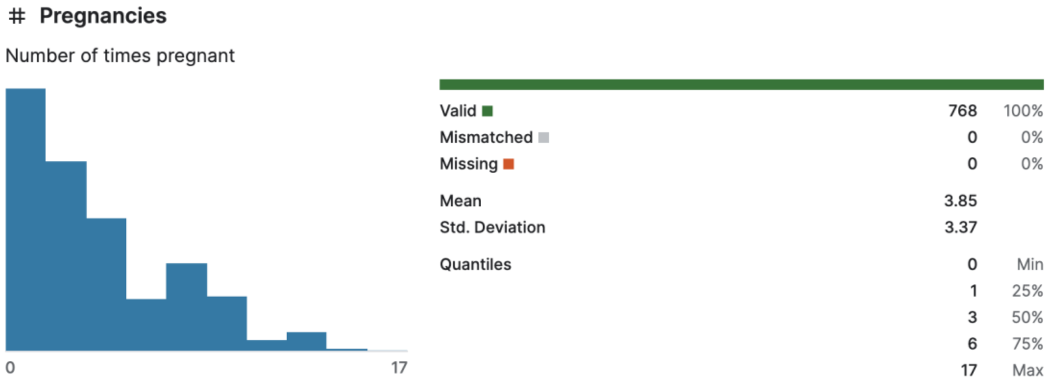
Age

Age (years)

Valid 🟩	768	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	33.2	
Std. Deviation	11.8	
Quantiles	21	Min
	24	25%
	29	50%
	41	75%
	81	Max

Pregnancies

The number of pregnancies is also a major factor, as they can cause gestational diabetes, and increase insulin resistance in the body. Mode is 0



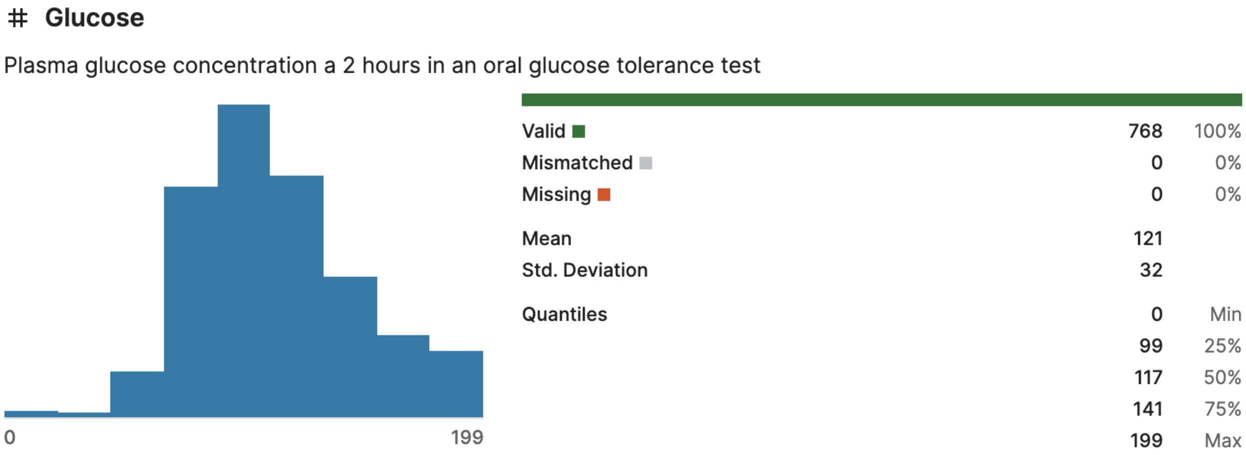
Pregnancies

Number of times pregnant

Valid 🟩	768	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	3.85	
Std. Deviation	3.37	
Quantiles	0	Min
	1	25%
	3	50%
	6	75%
	17	Max

Glucose

Average glucose levels of surveyed individuals. It is important to note that people who have a blood glucose concentration of more than 150. mg/Dl, have diabetes. And people between 110-148 are in the pre-diabetic spectrum. Mode 110



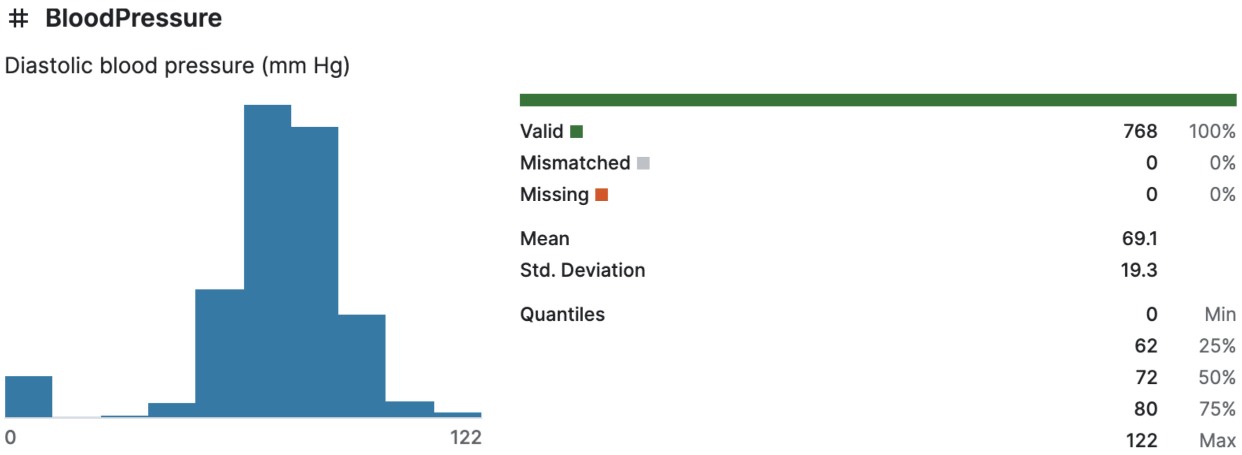
Glucose

Plasma glucose concentration a 2 hours in an oral glucose tolerance test

Valid 🟩	768	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	121	
Std. Deviation	32	
Quantiles	0	Min
	99	25%
	117	50%
	141	75%
	199	Max

Blood Pressure

High blood pressure (hypertension) can lead to many complications of diabetes, including diabetic eye disease and kidney disease, or make them worse. Most people with diabetes will eventually have high blood pressure, along with other heart and circulation problems. Mode 65



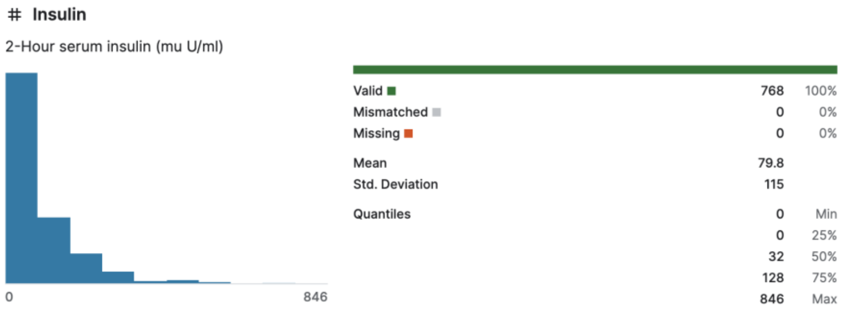
BloodPressure

Diastolic blood pressure (mm Hg)

Valid 🟩	768	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	69.1	
Std. Deviation	19.3	
Quantiles	0	Min
	62	25%
	72	50%
	80	75%
	122	Max

Insulin

Low levels of insulin are one of the leading causes of diabetes.



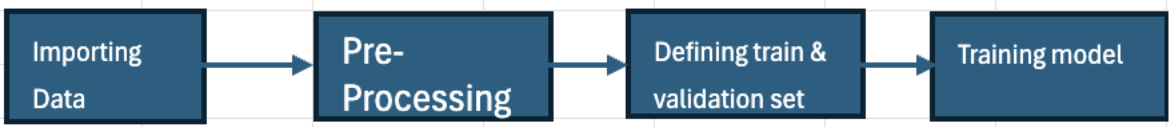
Insulin

2-Hour serum insulin (mu U/ml)

Valid 🟩	768	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	79.8	
Std. Deviation	115	
Quantiles	0	Min
	0	25%
	32	50%
	128	75%
	846	Max

Training the model

These are the broad steps that are followed for training any sort of model.



Importing Data → Pre-Processing → Defining train & validation set → Training model

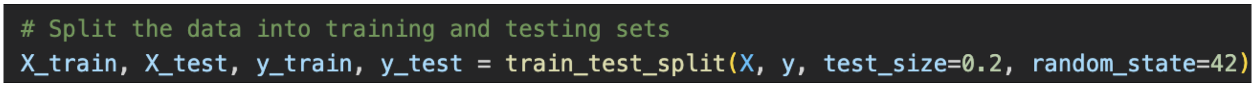
Pre-processing of data

After importing the data, it is always a good idea to check for any missing or “bad” values as they are said before training the data. Sometimes you can discover missing values in the dataset.

For any sort of missing data, I filled them with the mean (average) of the values of the same column. Sometimes the names of the columns also need to be changed as is the case with this dataset.

Defining the train and validation set

After pre-processing, we can start defining the test and training set from the dataset. This is done using this command in Python (scikit learn):

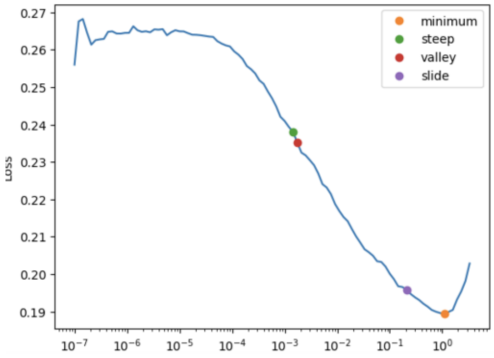


```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

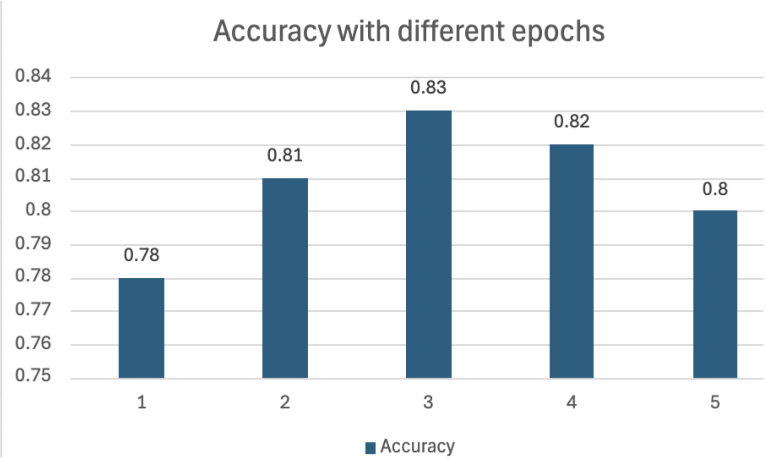
Here, 20% (0.2) of the dataset is assigned to the testing/validation set.

Training the model

To get the best possible results, I trained many different models using Linear, Logistic Regression (Using XGBoost and Fast.AI) and Random Forests. Before we move forward, I would like to answer one question that might have come into your minds. Below is a sample graph of what auto-generated learning rate suggestions look like.



I also experimented with many different epochs. Their accuracy I have put here in a graph. All the models were run with the same 0.05 learning rate.



[1: 1 | 2: 10 | 3: 30 |4: 40 | 5: 50] (Number of epochs)

What is XGBoost?

It stands for extreme gradient boosting, and what it essentially does is it improve the accuracy of the model by using many weak decision trees, improving upon them, and then combining them to create one powerful model. It can be used with the standard scikit learn protocol to train models from scratch. It has many parameters which need to be adjusted before you can train it. For instance, the learning rate, the number of trees (n_estimators), the kind of 'booster' along some more.

Results

I started off training the model with the Fast.AI library, which is a simplified way of training models. Its main drawback was that it can only do linear regression with its regression functionality and its accuracy was decent (88% on the validation set) but I wanted to increase the accuracy, so I tried out Random Forests, which reduced the accuracy. So, I used the XGBoost library to carry out boosted random forests, logistic, and linear Regressions. After changing many parameters and doing a lot of trial and error, I found the models giving the best accuracy. In the end, I ended up with a model producing close to 97% accuracy on the overall set. Overall, it took me over a week to finalize and decide the model to use.

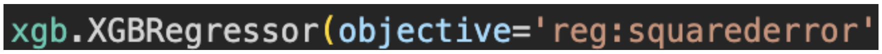
In case you are wondering, the most influential hyperparameters from my research are:

1. Learning Rate (learning_rate)

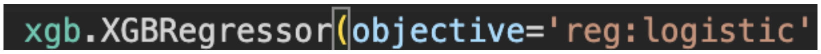
2. Number of trees (n_estimators)
3. Tree depth (usually due to the dataset running low on data) (max_depth)
4. The kind of XGB- you want to run, XGBClassifier or XGBRegressor

And finally

5. The objective (in the case of regression)



```
xgb.XGBRegressor(objective='reg:squarederror'
```

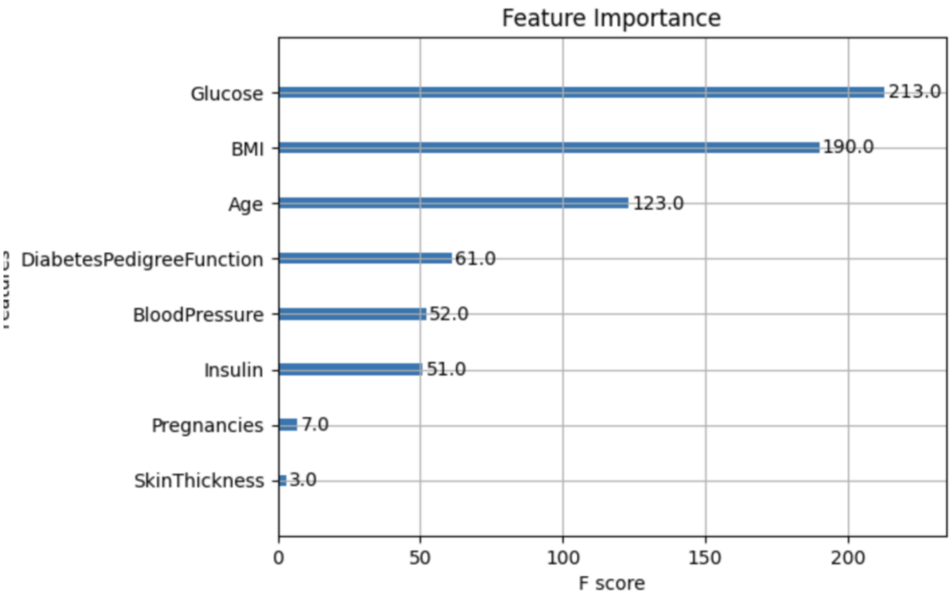


```
xgb.XGBRegressor(objective='reg:logistic'
```

The first one is used for linear regression, while the second one is used for logistic.

Feature Importance

What feature importance refers to is, how much "weightage" a model is giving to the columns. This is for the XGBoost model



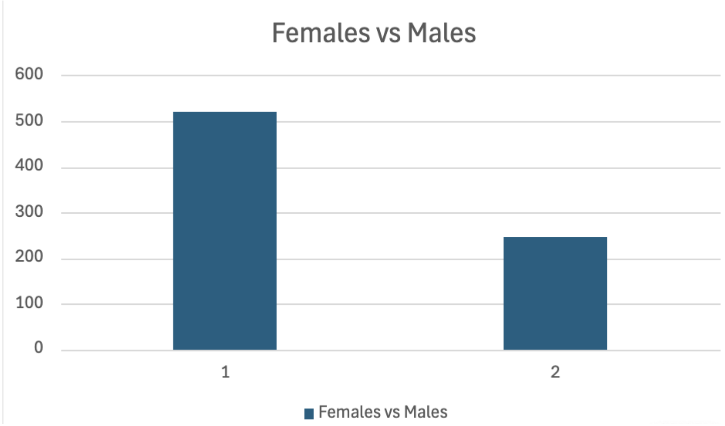
The Glucose, BMI and Age, are the highest importance factors, what that essentially means is that if glucose amount is high. > 150 in the blood and the BMI is also high, then, for certain the model will predict for the person to have diabetes. If it is a combination of these like a high BMI but mid-high glucose, the model mostly will predict a prediabetic.

Bias Analysis

I ran some experiments on how well the model performs on different people like, males or females. The accuracy for females is slightly better than the one for males. It might be because there is more data on females, rather than males in the dataset. 521 vs 246. After checking,

Females' predictability accuracy: 93%

Males' predictability accuracy: 92%



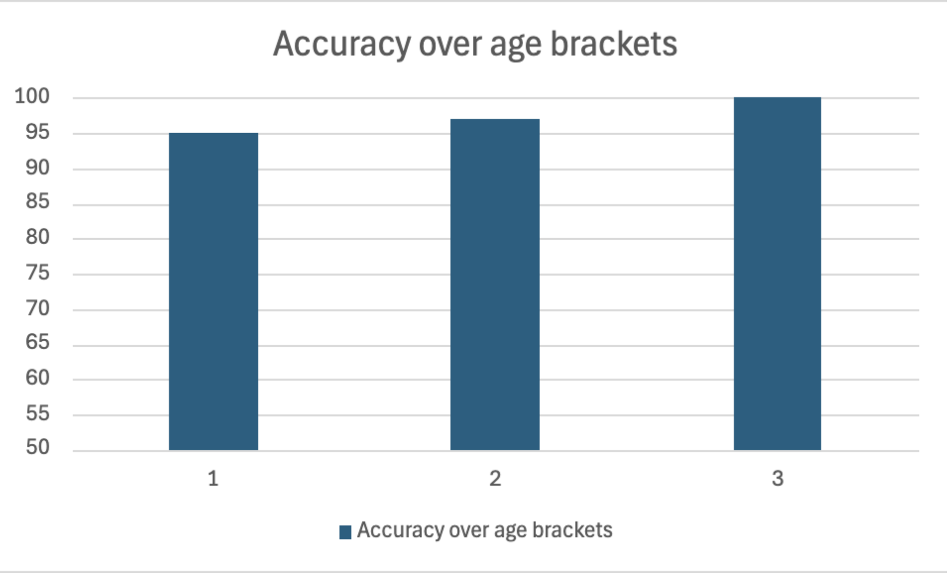
[1: Females | 2: Males]

Age bias analysis

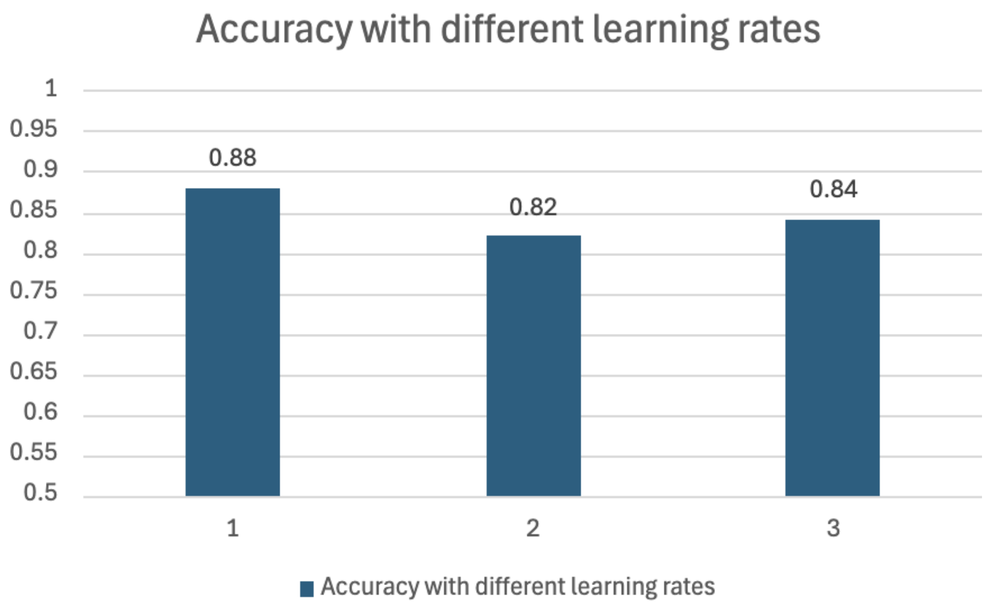
Age 20-30: **95.5% Accuracy**

Age 30 – 60: **97.1% Accuracy**

Age 60 – 81: **100% Accuracy**



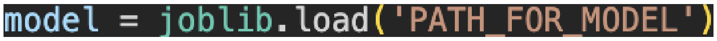
Learning rates



[1: 0.05 def | 2: 0.5 | 3: 0.005]

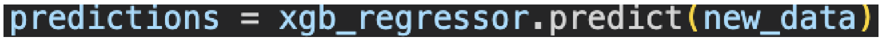
How to use the model

The model can be imported through the following command:



```
model = joblib.load('PATH_FOR_MODEL')
```

And predictions can be made through:



```
predictions = xgb_regressor.predict(new_data)
```

A user can refer to the 'Outcome' column filled, to check the probability of any one being diabetic. If multiple are filled, the 'predictions'- if called, will return a dictionary of all the predicted probabilities.

Sample usage scenario

The majority of diabetes cases are reported in developed countries. So, the implementation of such a system should not be difficult. Let's say you went to the clinic; you had your monthly checkup. In which, a sample of your blood was tested, your BMI, etc. This all and some general information (pregnancies, Diabetes Pedigree, etc.) can be passed into the model by a doctor, to tell if someone is at the risk of having diabetes or not. If you are a prediabetic, then they can advise you of the necessary steps and precautions.

Limitations

Given some very extreme values, sometimes the model fails to predict accurately.

Conclusion

We have discussed the basics of diabetes, why it poses a risk to people, what I have done to help, how the data was taken, some Exploratory Data Analysis, How to train a very high accuracy from scratch, looking inside the model and finally we ended with what the doctors do to help their patients get out of the diabetic spectrum and how easy the usage of such a model can be.

Acknowledgements

I would like to say a special thank you to my uncle, Dr. Shadab. Khan for introducing me to the field and helping me to learn and expand my knowledge of AI, Deep Learning and ML and always providing valuable feedback for my assignments and projects. I would also like to thank all the other people who have supported me on this journey.

References

World Health Organization

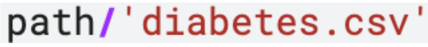
National Health And Nutrition Examination Survey (NHANES)

International Diabetes Foundation (IDF)

Appendix

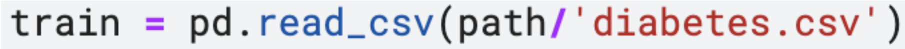
Handbook for each function used in code

Path: from pathlib. The code is used to shorten the amount of typing required when entering a new path of some file. You can assign a pre-designed directory to path. Then it can be called through:



```
path/'diabetes.csv'
```

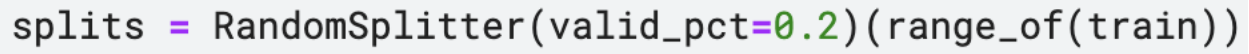
Pd.read_csv: from pandas. This function reads a csv from a directory. This step is necessary before a csv is passed in for any further processing or training. Called through:



```
train = pd.read_csv(path/'diabetes.csv')
```

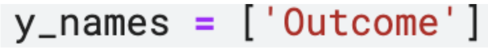
If the variable is called, the datasets sample is also given.

RandomSplitter: from fast.ai, (modified scikit-learn). Requires 2 arguments. The percentage of splitting of the original dataset and the dataset. This splits the dataset into training and validation set. Called through:



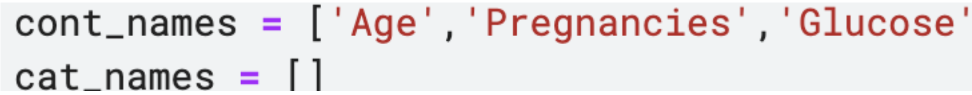
```
splits = RandomSplitter(valid_pct=0.2)(range_of(train))
```

Y_names: The target variable; the column you want to predict. A column name needs to be passed in.



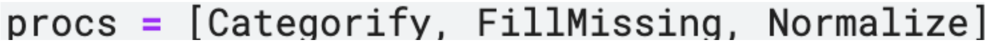
```
y_names = ['Outcome']
```

Cont_names and Cat_names: Names of the continious and categorical variables. Continious are like 1.2, 1.4, 5.7, etc… While categorical are like Bird or not. Sample use:



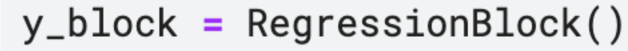
```
cont_names = ['Age','Pregnancies','Glucose'
cat_names = []
```

Procs: The processes you want to fill in. Requires 3 default arguments (for training on csv's)- Categorify, FillMissing and Normalize



```
procs = [Categorify, FillMissing, Normalize]
```

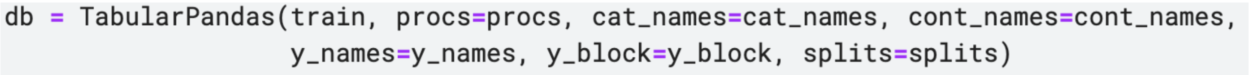
Y_block: What kind of action you want to model to do. Like: RegressionBlock, Vision_learner, etc.



```
y_block = RegressionBlock()
```

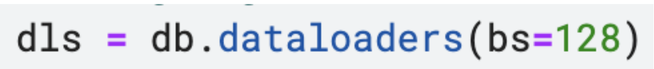
Note: from y_names – procs, all the arguments need to be in dictionaries

TabularPandas: Used for training of models on tabular data. Y_names – Y_block need to passed in as arguments, what this function does is that it creates the framework for training of the model.



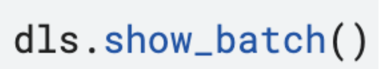
```
db = TabularPandas(train, procs=procs, cat_names=cat_names, cont_names=cont_names,
                   y_names=y_names, y_block=y_block, splits=splits)
```

Dataloaders: This function allows the user to assign the amount of data loaded in each turn. it requires 1 argument and that is of the batch size. This is very helpful for models being trained on small GPU's.



```
dls = db.dataloaders(bs=128)
```

Show_batch(): Allows user to see the data passed for training.

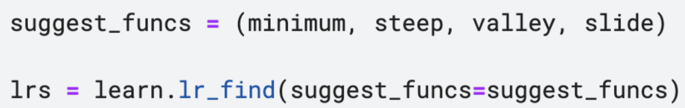


```
dls.show_batch()
```

Y_range: Can be used to allow to range of prediction values given.

Tabular_learner: the learner used for training on csv's. Some arguments required are the data, layers, metrics, config and the loss_function.

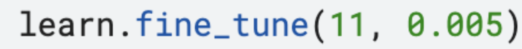
Lr_find: helps to find the best learning rate. Requires 1 argument, which describes what points at the graph are wanted (the argument needs to be pre-defined with a variable). (The learning rate graph in the report was generated through this command)



```
suggest_funcs = (minimum, steep, valley, slide)

lrs = learn.lr_find(suggest_funcs=suggest_funcs)
```

Fine_tune: This is the command which trains the model on the data. You need to pass in 2 arguments, which tell the learning rate and the number of epochs you want the model to do. 'Fit_onecycle' and 'fit' also do the same function



```
learn.fine_tune(11, 0.005)
```

.squeeze(): The .squeeze() function is used to remove single-dimensional entries from the shape of an array. It returns an array with the same data but reshaped dimension.

Test_dl: fast.ai function, which is used to test the trained model on new data. It requires 1 argument, the path of the new data.

.to_csv: a pandas function which exports the data changed, into a csv.

.export(): A fast.ai function, to export the trained model.

.iloc[]: A function of pandas, which allows the user to target/direct to specific columns. It requires 1 argument.

Train_test_split: The function splits the dataset into train and test sets. It requires 3 arguments, the feature columns, target column and the percentage to split. It has an optional random seed argument.

XGBClassifier & XGBRegressor: These are the methods in XGBoost which do as the name suggests. XGBClassifier does classification problems while the latter does regression problems. These require many arguments known as hyper-parameters.

XGboost Hyperparameters:

XGboost has many different hyperparameters here are some's default values and their names:

- **Booster**: This has 2 types: “gbtree” or “gblinear” Default is “gbtree”, gbtree uses gradient boosted trees, and if you select this, the model pretty much becomes a gradient boosted random forest.
- **N_estimators**: These are the number of trees in the XGBoost forest. Default 100
- **Learning_rate:** You need to pass the desired learning rate here. Default 0.3
- **Objective:** In objective, you can pass either reg: logistic, reg: linear, binary: logistic. Etc... Default is reg: logistic
- **Max_depth:** The maximum number of branches can be controlled here. Default is 6
- **Min_childweight:** The minimum amount of data needed before making a new branch. Helps prevent underfitting.

.predict(): Using this a model can predict a new data. Requires 1 argument and that of the path of the data.

Joblib.dump(): Exports the model, requires 2 arguments, the variable name of the model to export and the name someone wants of the exported model.

Plt.figure(): Part of matplotlib library, in this it requires 1 argument and that of the desired size of the graph.

Xgb.plot_importance(): Part of the XGBoost library and requires 2 arguments, the model and the metric on which to report. This is used for checking the feature importance in a model.

Plt.title(): You can input the title name for the graph. 1 argument.

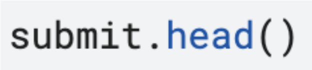
Plt.show(): Shows the graph.

Joblib.load(): Loads the model, requires 1 argument – the path of the model.

. drop(columns=["COLNAME”]): Drops the columns whose names are listed in the COLNAME. Requries 1 argument.

Submit[‘Colname’] = preds: Let’s assume that you have a csv file called submit and that you want to add the column name called colname in it. And you have the data and you want to enter that data. This is useful when entering predicted probabilities into a csv file.

.head(): Shows the top 10 rows in the dataset required. Needs to be passed in like:



```
submit.head()
```