EARLY STAGE DIABETES PREDICTION

Dataset

- Downloaded from the UCI Machine Learning Repository at:

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

- CSV (Comma Separated Value) file.

- Seems to be a more credible source than other websites such as Kaggle.

Use Case

- To create machine learning and deep learning models to predict early stage diabetes. These models can then be used in the healthcare industry to detect if someone could have early stage diabetes, which can then be verified by medical tests.

- Link to Jupyter Notebook:

https://github.com/davidehosein/Early-Stage-Diabetes-Prediction/blob/main/diabetes_prediction.ipynb

Architectural Choices

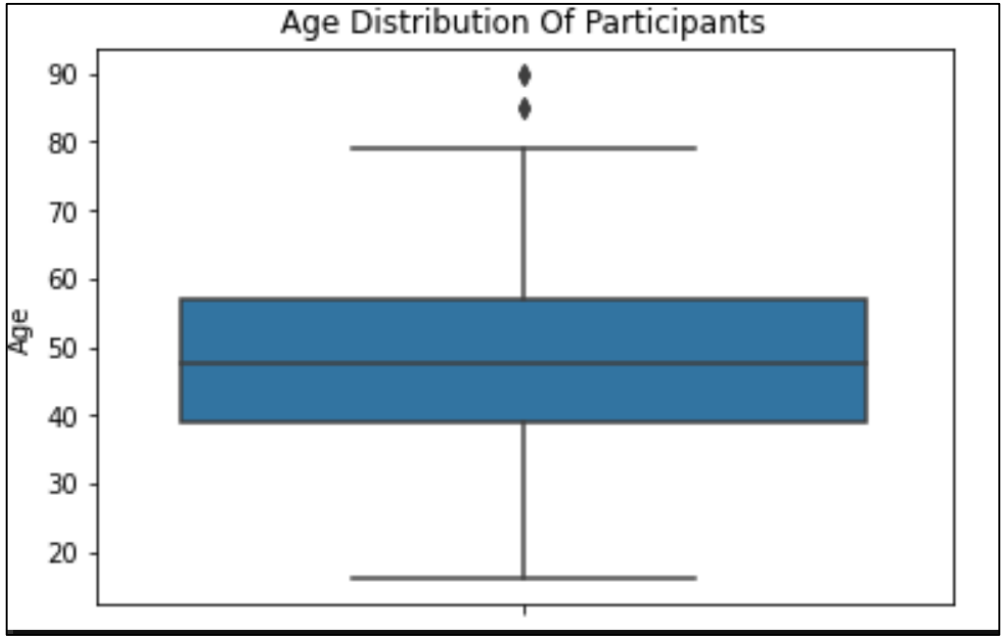
- Jupyter Notebook
- Programming Language: Python
- Python Libraries:
  - Pandas (For data cleaning, manipulation and analysis)
  - Matplotlib (For data visualization)
  - Seaborn (For data visualization)
  - Scikit-Learn (For creating machine learning models)
  - Keras (For creating deep learning models)

Data Quality Assessment

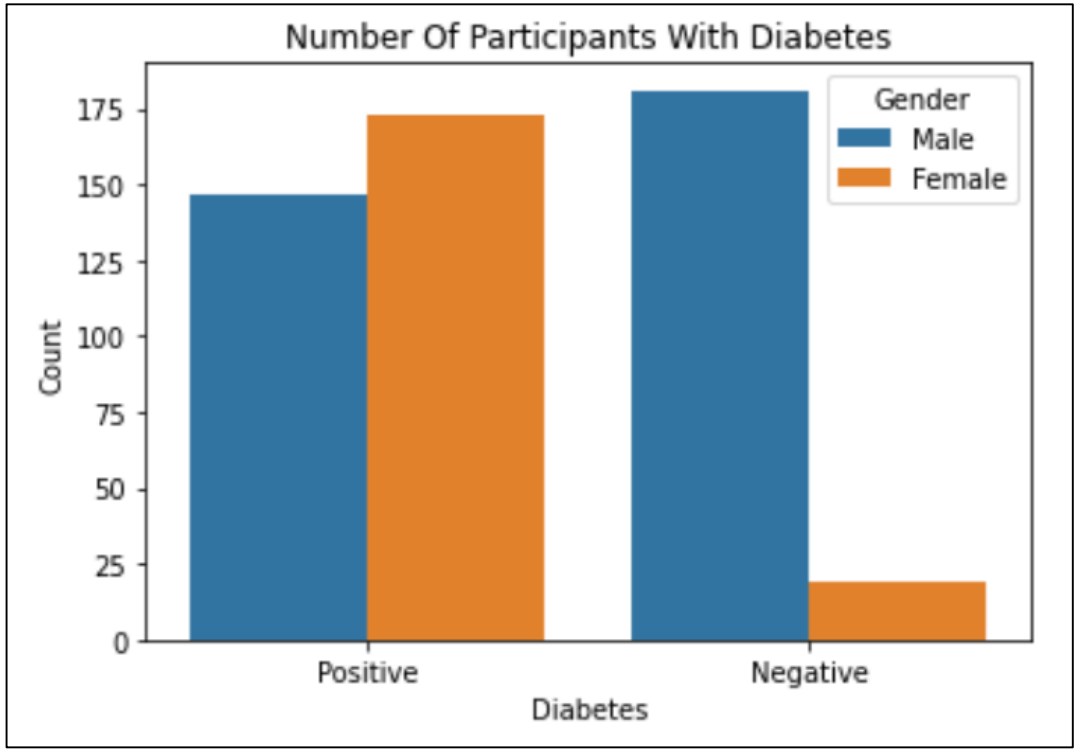
- No missing values
- Data already in a presentable format for analysis, visualization and modelling
- CSV file renamed from “diabetes_data_upload.csv” to “diabetes_data.csv”
- The column titles were converted to title case for consistency
- 17 columns
- 521 rows (including header)
- 16 columns containing categorical values
- Target column – Class (“Positive” or “Negative”)

Exploratory Data Analysis (EDA)

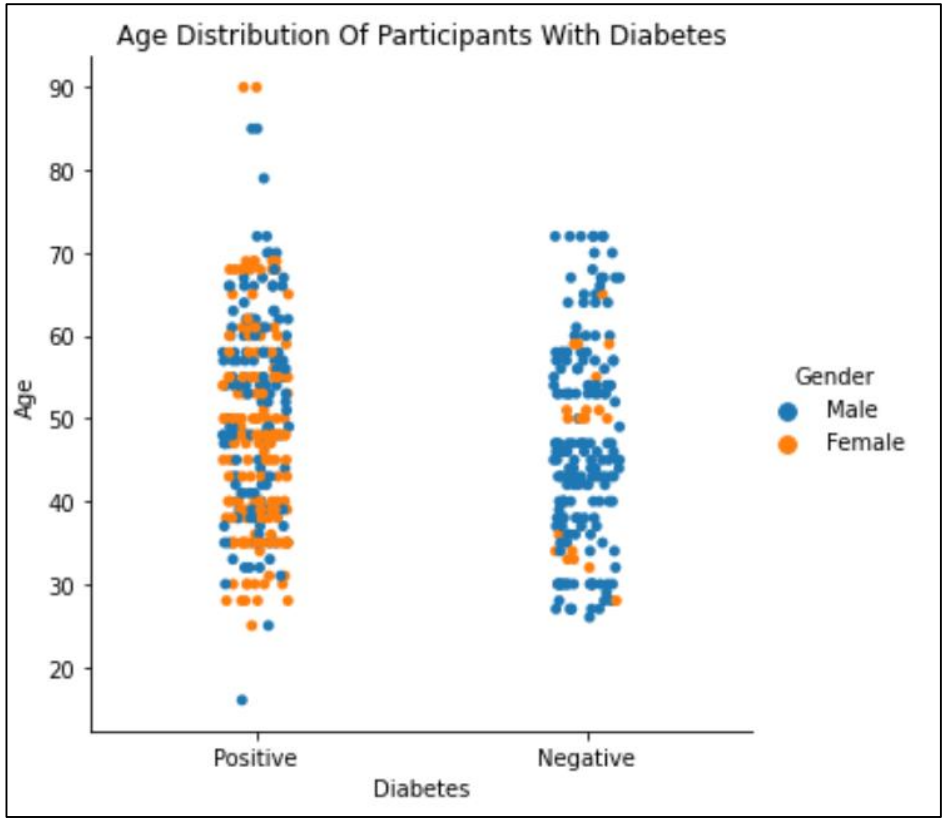
- Boxplot shows that most of the participants are between 16-80 years with a median age of 47.5 years. There are a few participants above 80 years.



- Bar graphs show that 320 out of the 520 participants have diabetes.
- Out of the 320 participants with diabetes, about 150 are males and 170 are females. It is possible that some of the females with diabetes could have developed it from Gestational Diabetes (GDM).



- Categorical scatter plot shows that the ages of participants with and without diabetes are around the same range. However, there are some elderly people (> 75 years) and young people (< 20 years) with diabetes.
- The relationship between age and diabetes cannot be distinguished.



Feature Engineering

- Models cannot work with categorical values.
- The categorical values in the target (y) column were converted into numerical values.
- “Positive” was converted to 1 and “Negative” was converted to 0.
- The first 16 columns in the dataset were selected as the features (X) for the models.
- 15 of the feature columns contained categorical values. These columns were converted into numerical feature columns using the "One Hot Encoding" technique.

Data Modelling

- Data Standardization using StandardScaler to scale the features between 0 and 1.
- Data was split into training and testing sets.
- 80% for training, 20% for testing.

- Models selected:
  - K-Nearest Neighbors (without hyperparameter tuning)
  - K-Nearest Neighbors (with hyperparameter tuning)
  - Simple deep learning model

- GridSearchCV (with 10 cross validations) was used to create the KNN model (with hyperparameter tuning).

- Hyperparameters selected for KNN model:
  - n_neighbors = list(range(1, 11))
  - algorithm = ['auto', 'ball_tree', 'kd_tree', 'brute’]
  - p = [1,2]

- Best Hyperparameters for KNN model:
  - n_neighbors = 1
  - algorithm = ‘auto’
  - p = 1

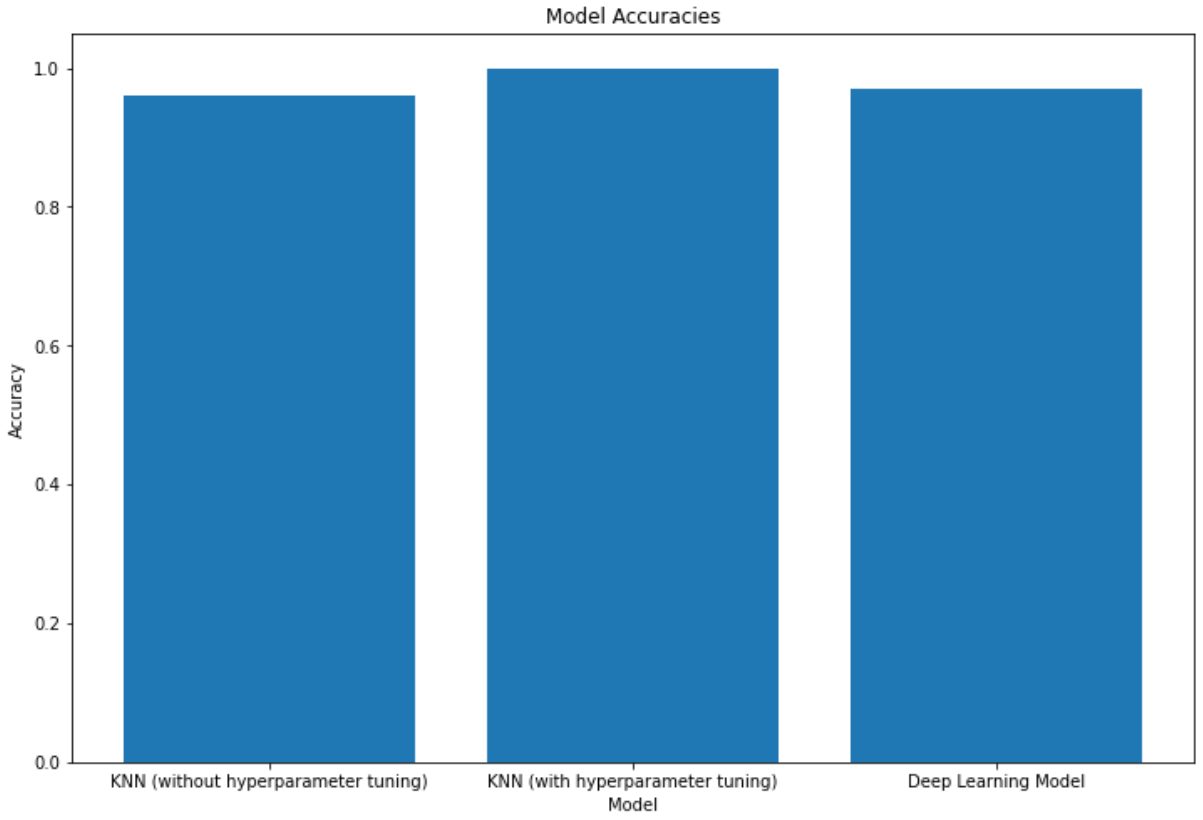
- Simple Deep Learning model:

```
deep_learning_model = Sequential()


deep_learning_model.add(Dense(12, input_dim=31, activation="relu"))
deep_learning_model.add(Dense(8, activation="relu"))
deep_learning_model.add(Dense(1, activation="sigmoid"))


deep_learning_model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])


deep_learning_model.fit(X_train, y_train, epochs=100, batch_size=10)
```

Model Evaluation

- The accuracy of the KNN models (with and without hyperparameter tuning) was evaluated using the accuracy_score() method in the Scikit-Learn library.

- The accuracy of the simple deep learning model was evaluated using the evaluate() method of the model.



- All of the models performed well and had an accuracy over 95%.
- Accuracies –
  - KNN model (without hyperparameter tuning) – 96.15%
  - KNN model (with hyperparameter tuning) – 100%
  - Deep Learning model – 98.08%

Conclusion

- Models created:
  - KNN (without hyperparameter tuning)
  - KNN (with hyperparameter tuning)
  - Simple Deep Learning model

- KNN model (with hyperparameter tuning) had the highest accuracy out of the three models with an accuracy of 100%, followed by the simple deep learning model and the KNN model (without hyperparameter tuning).