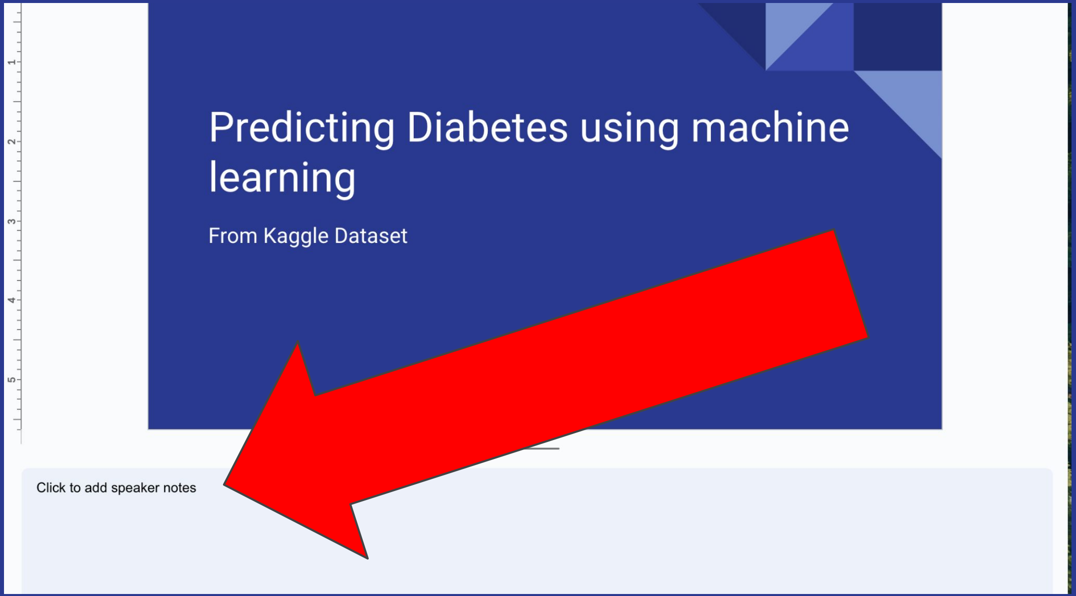
Predicting Diabetes using machine learning

From Kaggle Dataset

Author’s Note:

I will include all extra information in speaker notes, please open them for the rest of the presentation



Why Predict Diabetes?

- 1 in 4 adults have diabetes and don't know it
- It leads to heart failure and disease which is the leading cause of death of americans
- Almost 1% of all Americans have it

From:
https://www.cardiosmart.org/assets/infographic/diabetes

DIABETES
CardioSmart
American College of Cardiology
What is TYPE 2 DIABETES?
A condition that occurs when your body CAN'T PROPERLY PROCESS SUGAR INTO ENERGY.
The body fails to use insulin correctly, or
The pancreas fails to make enough insulin
More than 30 million adults in the U.S. have diabetes
About 1 in 4 adults with diabetes don't know they have it.
What are the SYMPTOMS?
Extreme thirst
Feeling hungry even while eating
Frequent urination
Slow-healing cuts
Numbness in hands or feet
Blurred vision
What Parts of Your Body Can BE AFFECTED BY DIABETES?
Nervous System
Eyes, Teeth & Gums
Coronary Arteries
Kidneys
Blood Vessels
Feet & Skin
Why is it DANGEROUS?
High blood sugar can:
Lead to stroke
Increase risk of heart disease or heart failure
Threaten vision, limbs & extremities
KEEP UP WITH HEALTH VISITS to find & treat problems early.
With help, YOU CAN CONTROL DIABETES.
Go to CardioSmart.org/Diabetes to learn more about making healthier choices.

Dataset: Kaggle Diabetes Dataset

Source: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Description:

- 768 Patient Records
- Columns include: Glucose levels, BMI, Age and more
- Target Variable: Outcome (1 = has diabetes, 0 = does not have diabetes)

Important step: I downloaded the dataset as a zip file from the website above.

Imports

Please copy these imports into your own IDE or notes

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report
import tensorflow as tf
from tensorflow import keras
from google.colab import files
```

Step 1: IDE: Google Colab

Setup: I opened a new notebook in Google Colab and uploaded my kaggle dataset like this:

```
uploaded = files.upload()
filename = list(uploaded.keys())[0]
data = pd.read_csv(filename)
```

Step 2: Printing the Data

Use the code below for the next couples slides.

```
# Step 2: Exploratory Data Analysis
print(data.info())
print(data.describe())
sns.pairplot(data, hue='Outcome')  # Visualize relationships
plt.show()
```

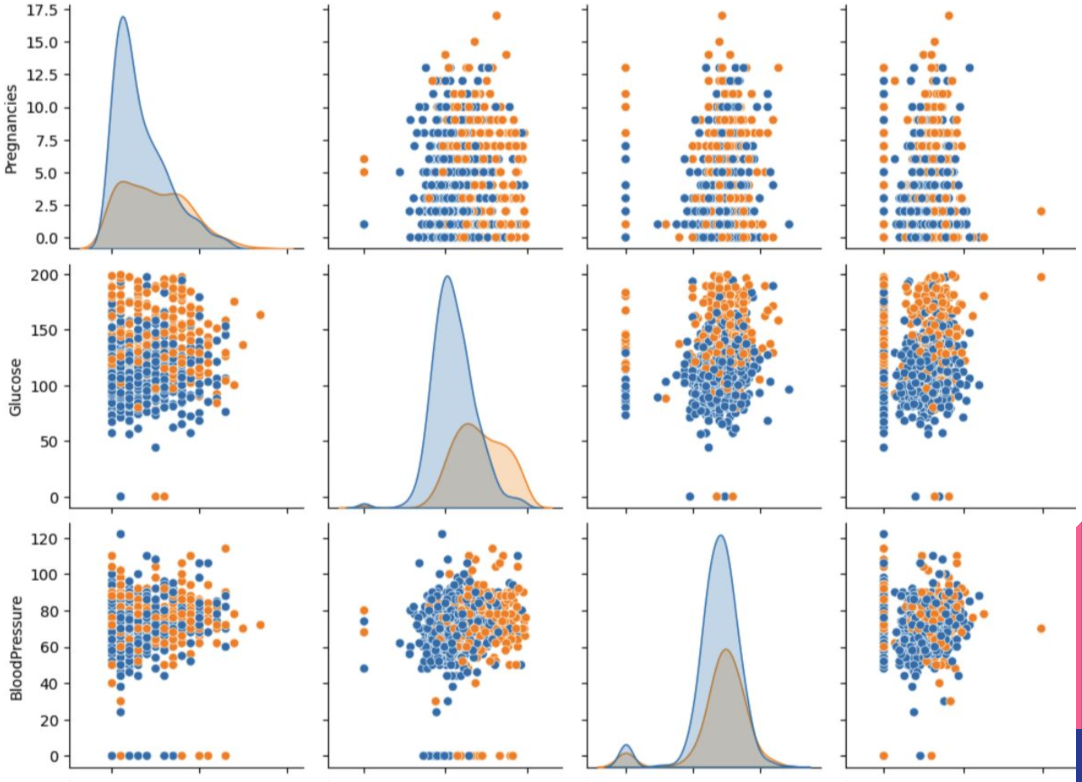
Looking at the data

Printed the data using data.info() and data.describe() with the code from the previous slide.

```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Data Analysis

Plot to see relationships on a very basic level, not required



Step 3: Clean the Data

Clean and wash the data before learning

```
# Handle missing values (replace 0s in key medical columns with NaN, then impute with mean)
cols_with_missing = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
data[cols_with_missing] = data[cols_with_missing].replace(0, np.nan)
data.fillna(data.mean(), inplace=True)

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

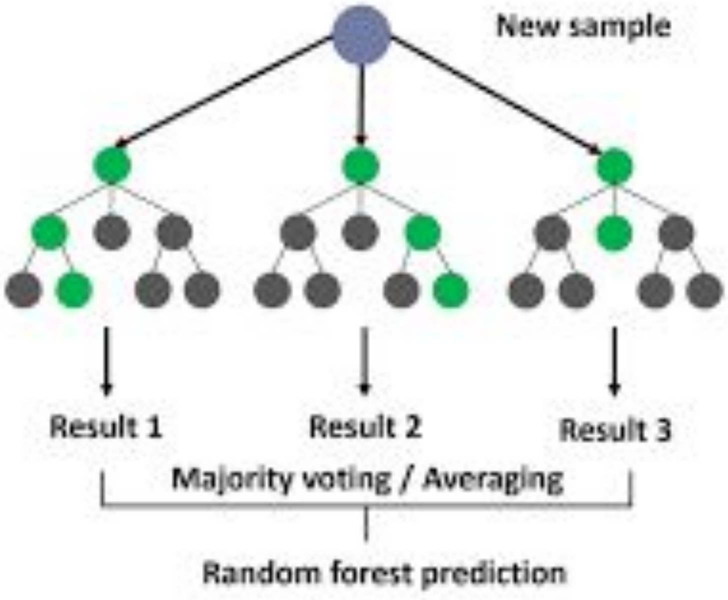
Step 4: Training the random forest model

Using the random forest model for training the datasets

```
# Step 4: Train a Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
print("Random Forest AUC:", roc_auc_score(y_test, rf_preds))
```

Random Forest Model Explained

Read the speaker notes for a more in depth breakdown of the random forest model



Step 5: Train a Deep Learning Model

Creating a deep learning model with sequential function.

```
# Step 5: Train a Deep Learning Model
model = keras.Sequential([
    keras.layers.Dense(16, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dense(8, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

Running the model

The code is to run the model with optimizer adam and binary cross entropy

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['AUC'])
model.fit(X_train, y_train, epochs=20, batch_size=16, validation_data=(X_test, y_test))
```

Step 6: Evaluate Deep Learning Model

Watching the model run, at this point you should see a lot of output from the computer, it is a good time to go for a coffee break because this can take some time. It shouldn't be over an hour though.

```
Random Forest Accuracy: 0.7272727272727273
Random Forest AUC: 0.703030303030303
Epoch 1/50
```

```
77/77 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - AUC: 0.9232 - loss: 0.3335 - val_AUC: 0.8109 - val_loss: 0.5256
```

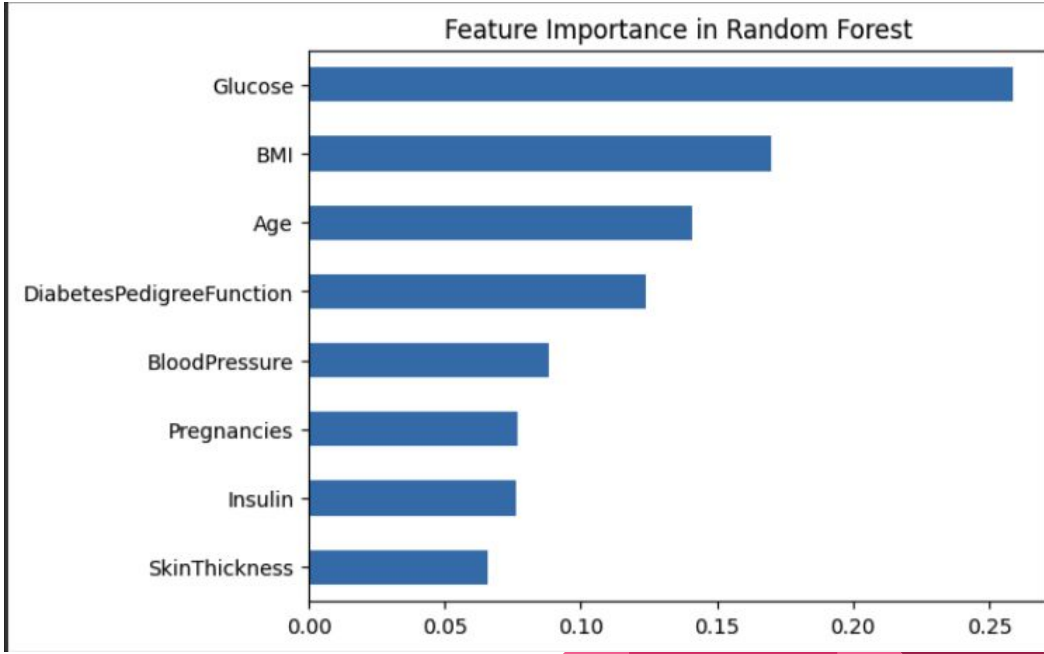
Step 7: Rank features in relation to diabetes

Once my model was run and my accuracy was closer to 95/100, I built a simple graph to compare the features or characteristics of diabetes.

```
# Step 7: Feature Importance
feature_importances = pd.Series(rf_model.feature_importances_, index=data.columns[:-1])
feature_importances.sort_values().plot(kind='barh')
plt.title("Feature Importance in Random Forest")
plt.show()
```

Conclusion: Who won or lost?

It appears that Glucose is the most important factor in determining diabetes presence in individuals followed by BMI and age. Surprisingly Insulin and Skin Thickness were on the lower end. So What does this mean in the real world?



Conclusion

After learning about how dangerous and common diabetes is, I built a model to try to differentiate the leading factors in order of importance to likelihood of having diabetes. I found that in almost 800 patients Glucose levels were the most important factor in predicting diabetes. Meaning that in the future of diagnosing diabetes, according to my model, we should look at glucose levels first and the most discriminately. Insulin is the least important and we should ignore or value the levels less. This might be contrary to popular belief as insulin is very much related to diabetes but levels of insulin do not predict diabetes as well as glucose levels do.

Thank you for watching this tutorial!

Please leave some feedback for me to make improvements!