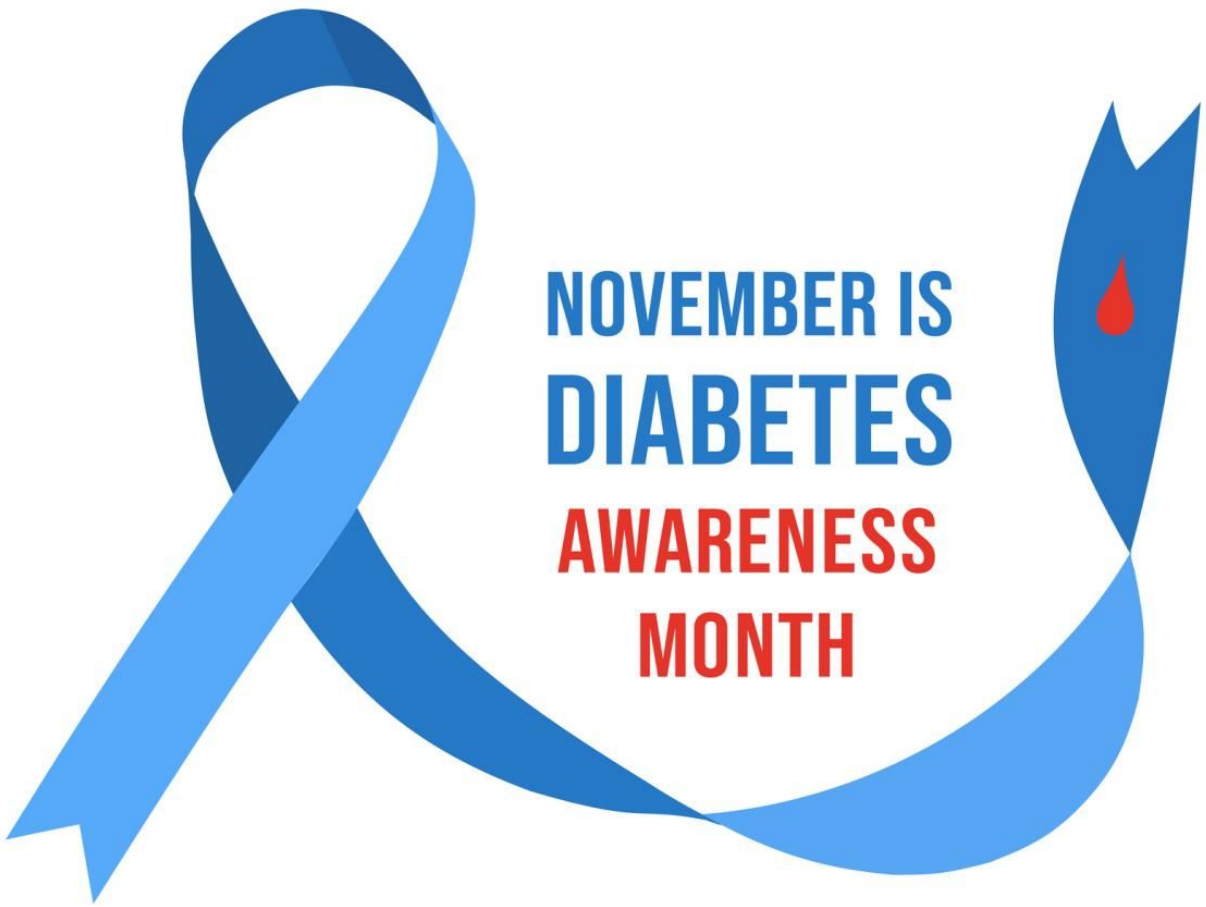
Can Machine Learning Accurately Predict Diabetes?

Analysis by: Leah Latham, Adrian Sandoval, Tiffany Conrad, Damla Duman, and Daniel Meyerowitz


NOVEMBER IS
DIABETES
AWARENESS
MONTH

1 in 5

About 38 million Americans have diabetes, and 1 in 5 don't know it.

CDC 2023
U.S Department of Health and Human Services

First, we had to find a robust dataset. Kaggle had us covered!

```
# VALUE_COUNTS FOR DIABETES CATEGORIES (0) - no diabetes (1) - pre-diabetes (2) - diabetes
dm_df["Diabetes_012"].value_counts()
```

```
0.0    213703
2.0     35346
1.0      4631
```

Data Ranges that Need Further Explanation

- Age-Categorized the data into 5 years ranges , starting from age of 18
- BMI- 18.5-24.9= Healthy Weight range, 25.0-29.9 = overweight range, >30 = obese range
- Income- 1-8 scale. 1 = less than $10,000, 5 = less than $35,000, 8 = $75,000 or more
- Education- 1 = none-Kindergarten, 2 = grades 1-8, 3 = grades 9-11, 4 = grade 12-GED, 5 = 1-3 years college, 6 = 4+ years of college
- PhysHealth - How many days in that month have you been concerned about your physical health? (0-30)
- GenHealth - How many days in that month have you been concerned about your general health? (0-30)
- MentalHealth - How many days in that month have you been concerned about your mental health? (0-30)
- Sex - 0 is female, 1 is male

22 Columns Total, the rest of which are binary (0 for no, 1 for yes)

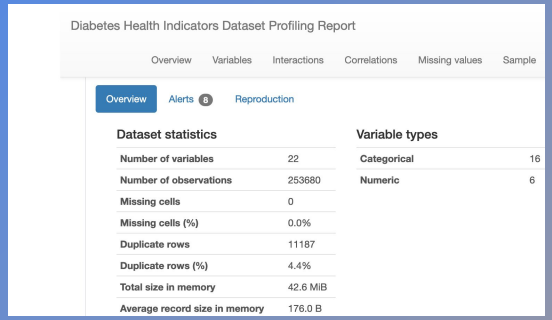
Data Cleaning (luckily, not too much!)

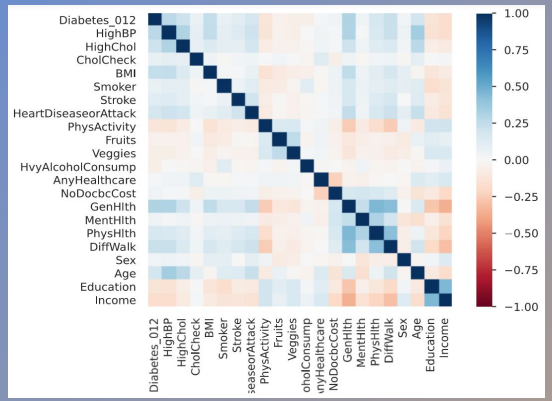
Cleaning data

```
# CREATE COPY FOR CLEANING
di_df = dm_df.copy()

# CONVERTING DTYPES TO INT FOR EASIER DATA MANIPULATION
di_df = di_df.astype(int)

# SIMPLIFYING TARGET COLUMN TO ONLY 0 (no diabetes) AND 1 (diabetes)
di_df = di_df.loc[(di_df["Diabetes_012"] != 1), :]
di_df["Diabetes_012"] = di_df["Diabetes_012"].replace(2,1)

# SHOW DATAFRAME
di_df.head(10)
```

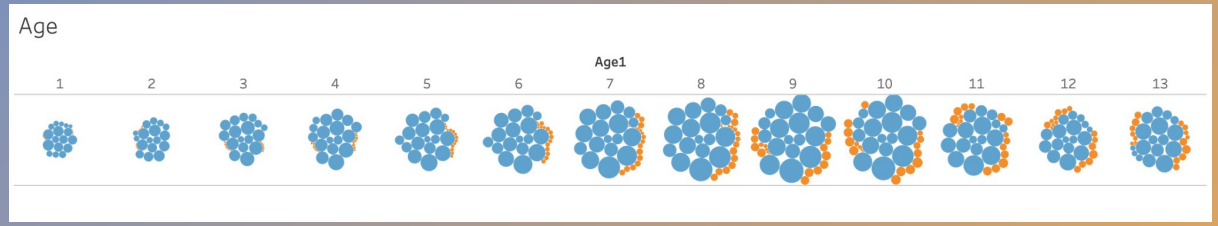
Exploration (more to come later!)

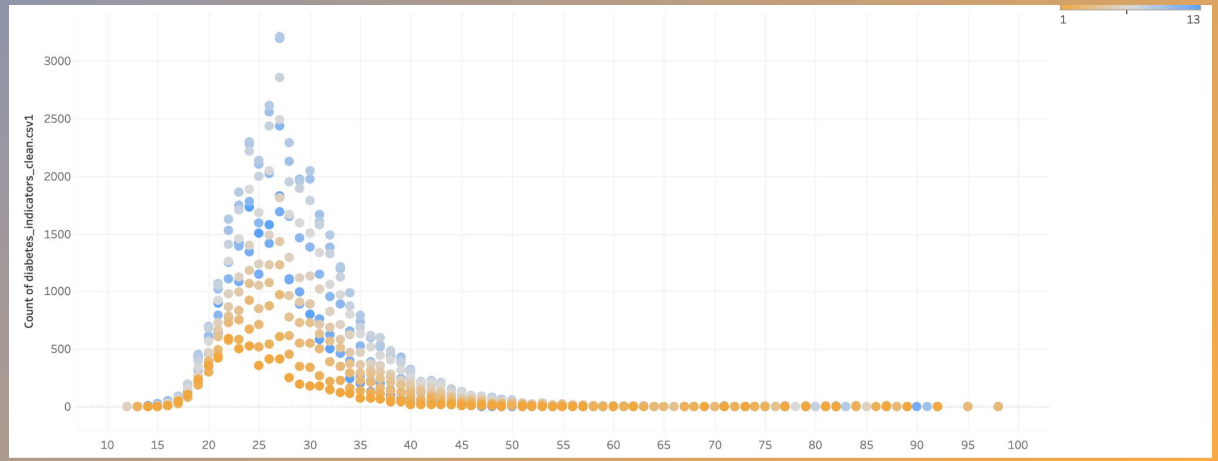
Ydata Profiling





Tableau





Machine Learning

```
1982/1982 - 1s - loss: -7.6734e+03 - accuracy: 0.5389 - 1s/epoch - 708us/step
Loss: -7673.44677734375, Accuracy: 0.538883626461029
```

NN_MODEL_1 - baseline results : `Accuracy 53.9%`

```
1982/1982 - 1s - loss: -1.2873e+04 - accuracy: 0.6861 - 1s/epoch - 686us/step
Loss: -12872.9365234375, Accuracy: 0.686092734336853
```

NN_MODEL_2 - reduced features results : `Accuracy 68.6%` (15% improvement from baseline)

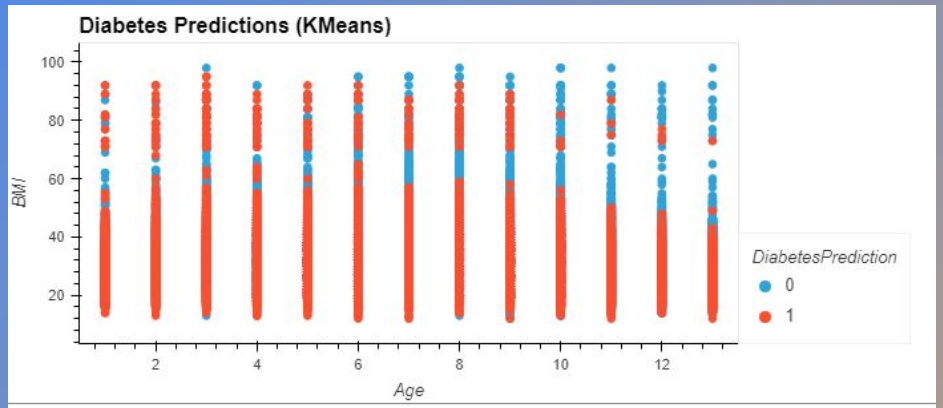
```
1946/1946 - 1s - loss: 0.3148 - accuracy: 0.8640 - 1s/epoch - 663us/step
Loss: 0.3148297965526581, Accuracy: 0.863980233669281
```

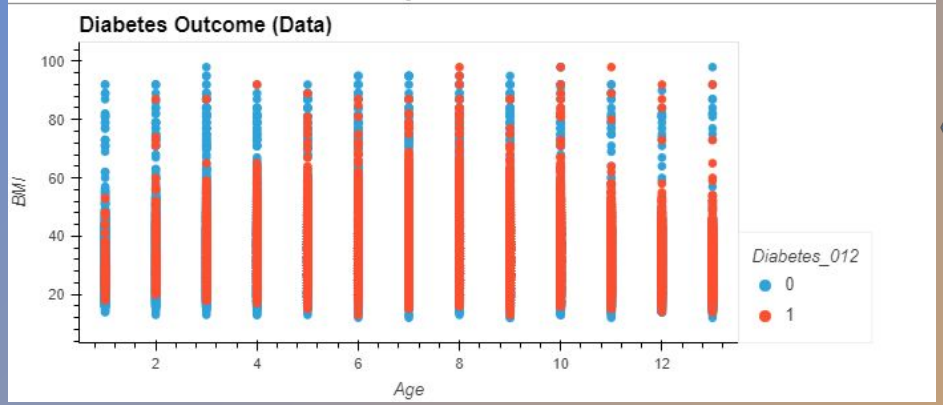
NN_MODEL_3 - cleaned data results : `Accuracy 86.4%` (+18% improvement from previous model)

```
1946/1946 - 2s - loss: 0.3164 - accuracy: 0.8644 - 2s/epoch - 826us/step
Loss: 0.3163652718067169, Accuracy: 0.8643656969070435
```

NN_MODEL_4 - best tuner results : Accuracy `86.4%` (minimal change from cleaned data)

Random Forests and KMeans





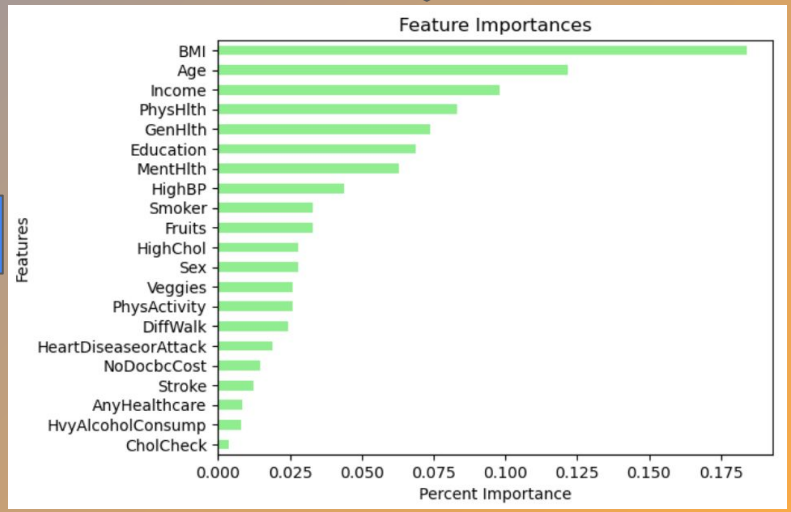
Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	51915	1608
Actual 1	7032	1708

Accuracy Score : 0.8612337985641553

Classification Report

	precision	recall	f1-score	support
0	0.88	0.97	0.92	53523
1	0.52	0.20	0.28	8740
accuracy			0.86	62263
macro avg	0.70	0.58	0.60	62263
weighted avg	0.83	0.86	0.83	62263



To see more of our data exploration…

Explore here ….