PREDICTING DIABETES

Sami Chowdhury, Yiannis Pagkalos, Lauren Christiansen, Mei Kam Bharadwaj , Dhwani Patel

Table of contents

EXECUTIVE SUMMARY

- Our **project goal** was to determine how key datapoints (BMI, high blood pressure, cholesterol, stroke, heart disease/attack, physical activity level, general health level, physical health level, difficulty walking scale, age, education level, income level) relate to the diagnosis of diabetes in different patients.
- We used this underlying data to create and train machine learning models to easily predict whether a patient would be diagnosed.

GOALS/ QUESTIONS

Our aim was to identify and utilize the most impactful datapoints (e.g., BMI, blood pressure, cholesterol, lifestyle factors) to improve our model's accuracy and reliability on predicting a diabetes diagnosis.

Questions:

1. Which dataset is best to train our model (number of features, rows and quality)?
2. Which features are most strongly correlated to diabetes diagnoses, and how do they contribute to the model's predictions?
3. What are the models that will produce the best accuracy scores?
4. How can we finetune the selected models to amplify accuracy scores?

DATA COLLECTION / CLEAN UP

- Collected at the following Kaggle link by Alex Teboul
- Obtained from the Behavioral Risk Factor Surveillance System (BRFSS), an annual telephone survey that is collected annually by the CDC.
- The features are either questions asked of participants or variables calculated based on their responses. The dataset includes the following:
  - ['Diabetes_binary', 'HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker', 'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age', 'Education', 'Income']

APPROACH

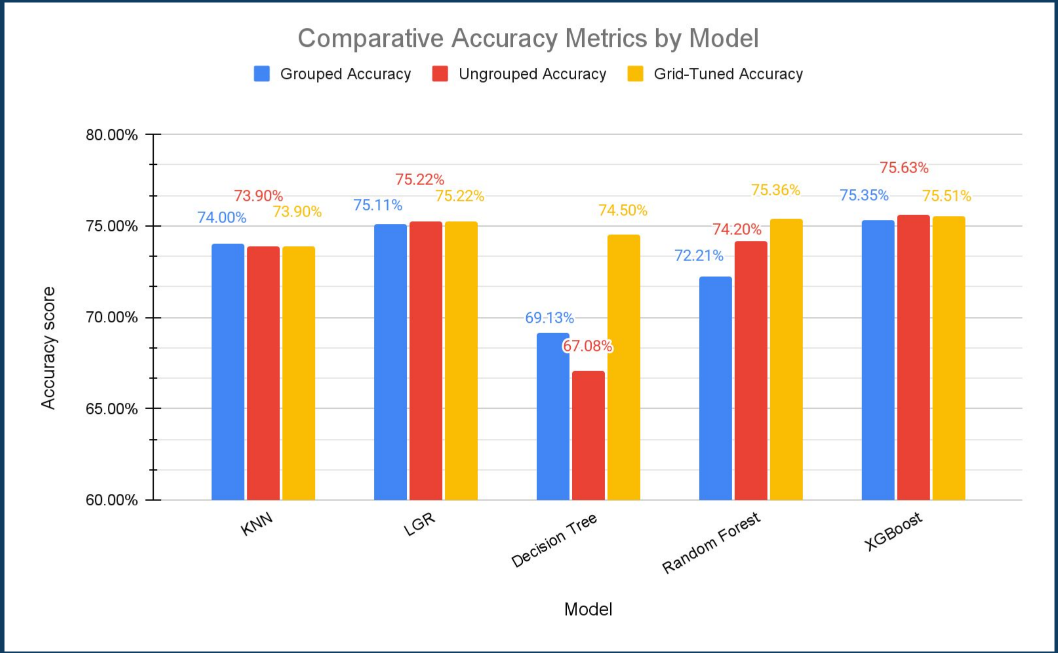
- Data was already balanced, cleaned for nulls (Keggle included both balanced and unbalanced versions of the data)
- Required encoding of a few set features (BMI condensed to scientific classifications from Underweight to Obesity 3, and initially also encoded AGE, INCOME, EDUCATION)
- In the end, we only encoded BMI and kept the original groupings of AGE, INCOME, and EDUCATION as this ended up producing better results
- Initially we did not remove features that were not strongly correlated to the results. Eventually removing those improved our model performance.
- Data was scaled since we had many ordinal encoders, all categorical and ordered
- Finally, we run GridSearchCV for all the models to further optimize performance by identifying model parameters that could help

RESULTS AND CONCLUSIONS

Given the nonlinear classification of the data, we experimented with the following models below. After all optimizations the following results were achieved:

- Decision Tree: 74.5%
- KNeighborsClassifier: 73.9%
- Logistic Regression: 75.2%
- RandomForestClassifier: 75.4
- XGBClassifier: 75.6%

RESULTS AND CONCLUSIONS



- The biggest improvements were achieved in the Decision Tree and the Random Forest model after the GridSearch optimization
- Interestingly, KNN and LGR models performed slightly worse after we ungrouped the Age, Education and Income categories.
- Similarly, XBoost performed better before GridSearch but only marginally.

DECISION TREE

	Predicted 0	Predicted 1
Actual 0	6240	2607
Actual 1	3246	5580

Accuracy: 0.67
Test score: 0.67
Train Score: 0.93

Classification Report

	precision	recall	f1-score	support
0.0	0.66	0.71	0.68	8847
1.0	0.68	0.63	0.66	8826
accuracy			0.67	17673
macro avg	0.67	0.67	0.67	17673
weighted avg	0.67	0.67	0.67	17673

No parameters specified

GridSearchCV

	Predicted 0	Predicted 1
Actual 0s	6336	2511
Actual 1s	1995	6831

Accuracy: 0.745
Test score: 0.745
Train Score: 0.745

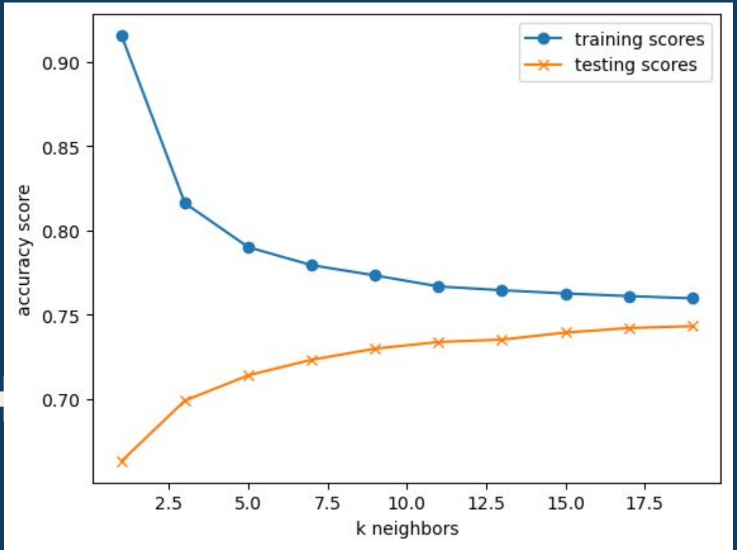
Classification Report

	precision	recall	f1-score	support
0.0	0.76	0.72	0.74	8847
1.0	0.73	0.77	0.75	8826
accuracy			0.75	17673
macro avg	0.75	0.75	0.74	17673
weighted avg	0.75	0.75	0.74	17673

```
{'max_depth': 7, 'min_samples_leaf': 1, 'min_samples_split': 2}
best score 0.738
```

- Specifying the additional parameters (max depth, min sample leaf and min sample split gives a significant improvement in accuracy: - (from 67% to 74.5%)

KNN MODEL



Confusion Matrix

	Predicted 0	Predicted 1
Actual 0s	6239	2608
Actual 1s	1997	6829

Accuracy: 0.739
Test score: 0.739
Train Score: 0.763

Classification Report

	precision	recall	f1-score	support
0.0	0.76	0.71	0.73	8847
1.0	0.72	0.77	0.75	8826
accuracy			0.74	17673
macro avg	0.74	0.74	0.74	17673
weighted avg	0.74	0.74	0.74	17673

- Training and Test scores converged as k increased
- k=15 seems the best choice for this dataset
- For k=15 accuracy= 73.9%

KNN MODEL

GridSearchCV

	Predicted 0	Predicted 1
Actual 0s	6239	2608
Actual 1s	1997	6829

Accuracy: 0.739
Test score: 0.739
Train Score: 0.763

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.76      0.71      0.73      8847
         1.0       0.72      0.77      0.75      8826

    accuracy                           0.74     17673
   macro avg       0.74      0.74      0.74     17673
weighted avg       0.74      0.74      0.74     17673

{'algorithm': 'auto', 'metric': 'euclidean', 'n_neighbors': 15, 'weights': 'uniform'}
best score 0.733
```

Additional parameter setting did not improve the model score (algorithm, metric, weights)
Max achieved accuracy - 73.9%

LOGISTIC REGRESSION

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0s	6513	2334
Actual 1s	2045	6781

Accuracy: 0.752
Test score: 0.752
Train Score: 0.745

Classification Report

	precision	recall	f1-score	support
0.0	0.76	0.74	0.75	8847
1.0	0.74	0.77	0.76	8826
accuracy			0.75	17673
macro avg	0.75	0.75	0.75	17673
weighted avg	0.75	0.75	0.75	17673

```
random_state=1, max_iter=100
```

GridSearchCV

	Predicted 0	Predicted 1
Actual 0s	6513	2334
Actual 1s	2045	6781

Accuracy: 0.752
Test score: 0.752
Train Score: 0.745

Classification Report

	precision	recall	f1-score	support
0.0	0.76	0.74	0.75	8847
1.0	0.74	0.77	0.76	8826
accuracy			0.75	17673
macro avg	0.75	0.75	0.75	17673
weighted avg	0.75	0.75	0.75	17673

```
{'C': 0.1, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'}
best score 0.744
```

- Additional parameter setting did not improve performance:
- Max Accuracy = 75.2%

RANDOM FOREST

	Predicted 0	Predicted 1
Actual 0s	6393	2454
Actual 1s	2112	6714

Accuracy: 0.742
Test score: 0.742
Train Score: 0.735

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.75      0.72      0.74      8847
         1.0       0.73      0.76      0.75      8826

    accuracy                           0.74     17673
   macro avg       0.74      0.74      0.74     17673
weighted avg       0.74      0.74      0.74     17673
```

`random_state=1, n_estimators=500, max_depth=3`

GridSearchCV

	Predicted 0	Predicted 1
Actual 0s	6351	2496
Actual 1s	1855	6971

Accuracy: 0.754
Test score: 0.754
Train Score: 0.776

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.77      0.72      0.74      8847
         1.0       0.74      0.79      0.76      8826

    accuracy                           0.75     17673
   macro avg       0.76      0.75      0.75     17673
weighted avg       0.76      0.75      0.75     17673

{'max_depth': 11, 'n_estimators': 1000}
```

- Increasing max depth to 11 from 3 and n_estimatros from 500 to 1000, resulted in a modest increase in accuracy (from 74.2% to 75.4%)

XGBOOST

	Predicted 0	Predicted 1
Actual 0s	6322	2525
Actual 1s	1782	7044

```
Accuracy: 0.756
Test score: 0.756
Train Score: 0.754
```

Classification Report

	precision	recall	f1-score	support
0.0	0.78	0.71	0.75	8847
1.0	0.74	0.80	0.77	8826
accuracy			0.76	17673
macro avg	0.76	0.76	0.76	17673
weighted avg	0.76	0.76	0.76	17673

```
xgb_model = XGBClassifier(random_state=1,
learning_rate=0.05, n_estimators=1000, max_depth=3)
```

GridSearchCV

	Predicted 0	Predicted 1
Actual 0s	6317	2530
Actual 1s	1798	7028

```
Accuracy: 0.755
Test score: 0.755
Train Score: 0.752
```

Classification Report

	precision	recall	f1-score	support
0.0	0.78	0.71	0.74	8847
1.0	0.74	0.80	0.76	8826
accuracy			0.76	17673
macro avg	0.76	0.76	0.75	17673
weighted avg	0.76	0.76	0.75	17673

```
{'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 100}
best score 0.748
```

- Interestingly, GridSearchCV results in a slightly less accurate model. Our initial parameter setting (learning rate = 0.05 vs 2 n_estimators = 1000 vs 100 and max depth 3 (same) give the best result from all our efforts: Accuracy of 75.6%

POTENTIAL NEXT STEPS

- We could explore more datasets that include health indicators such as HbA1C (hemoglobin A1C) and fast blood sugar test (FBS).
    - HbA1C test measures the average blood sugar (glucose) level over the past 60-90 days. A fasting blood sugar test measures the blood sugar levels first thing in the morning before the patient breaks their fast.
    - If the patient's blood sugar is high, then it indicates that patient has difficulties breaking down sugar in their body.
    - It is best to look at a dataset that includes both HbA1C and FBS data. HbA1C tests are less sensitive compared to the FBS test, but provides a more comprehensive story on the patient's blood sugar over a period of months.
    - In practice, **both** tests are used in the office to get a more accurate diagnosis of diabetes.

- We could experiment with more models and/or model parameter tuning

THANK YOU!

APPENDIX

DATA CORRELATION MATRIX

	Diabetes_binary
Diabetes_binary	1.000000
HighBP	0.381516
HighChol	0.289213
CholCheck	0.115382
BMI	0.293373
Smoker	0.085999
Stroke	0.125427
HeartDiseaseorAttack	0.211523
PhysActivity	-0.158666
Fruits	-0.054077
Veggies	-0.079293
HvyAlcoholConsump	-0.094853
AnyHealthcare	0.023191
NoDocbcCost	0.040977
GenHlth	0.407612
MentHlth	0.087029
PhysHlth	0.213081
DiffWalk	0.272646
Sex	0.044413
Age	0.278738
Education	-0.170481
Income	-0.224449

COLUMNS / COLUMN DESCRIPTIONS

Columns

In [24]:

```
diabetes.columns
```

Out[24]:

```
Index(['Diabetes_binary', 'HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
       'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
       'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth',
       'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age', 'Education',
       'Income'],
      dtype='object')
```

Column Descriptions

1. Diabetes_binary: 0 = no diabetes 1 = prediabetes or diabetes
2. HighBP: 0 = no high BP 1 = high BP
3. HighChol: 0 = no high cholesterol 1 = high cholesterol
4. CholCheck: 0 = no cholesterol check in 5 years 1 = yes cholesterol check in 5 years
5. BMI: Body Mass Index

BMI CLASSIFICATION

BMI

```python
def BMI_classification(x):
    if x < 18.5:
        return "Underweight"
    elif x > 18.5 and x <=24.9:
        return "Normal"
    elif x > 24.9 and x <= 29.9:
        return "Overweight"
    elif x > 29.9 and x <= 34.9:
        return "Obesity 1"
    elif x > 34.9 and x <= 39.9:
        return "Obesity 2"
    elif x > 39.9:
        return "Obesity 3"
```

```python
diabetes_df['BMI'] = diabetes_df['BMI'].apply(BMI_classification)
diabetes_df['BMI'].value_counts()
```

```
BMI
Overweight     24135
Obesity 1      17301
Normal         14460
Obesity 2       8112
Obesity 3       6031
Underweight      653
Name: count, dtype: int64
```