Predicting Diabetes

Project 4

(UTOR-VIRT-DATA-PT-06-2023-U-LOLC-MWTH(B))

Contributors (Group 4):

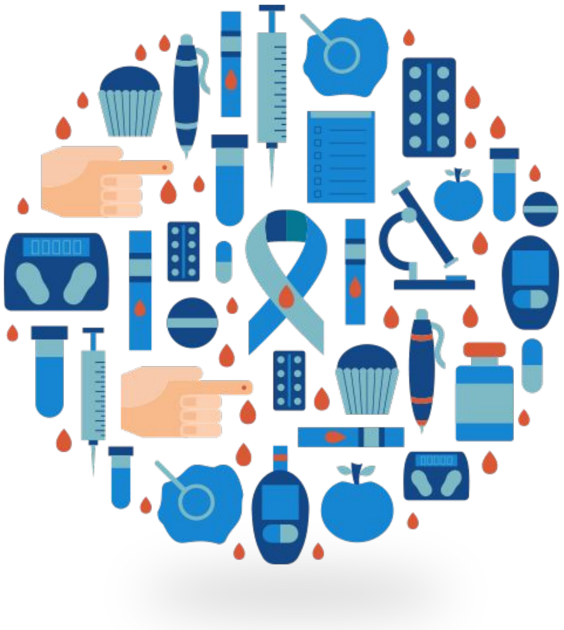
- Daiana Spataru
- Nikita Gahoi
- Priyanshi Gajjar
- Lydia Zuo
- Mohammad Islam

The Topic: Diabetes

- Diabetes affects the health of millions of people and puts an enormous financial burden on the US economy
- Early diagnosis of diabetes can lead to lifestyle changes and more effective treatment
- Predictive models for diabetes risk can be an important tool for public health officials.

The Goal

Develop a machine learning model to identify individuals that either have diabetes or are high-risk for developing diabetes using screening information.



Division of Work

Nikita

Data Collection & Transformation

Pandas
Numpy

→

Daiana

Database Setup

AWS RDS
pgAdmin 4

→

Lydia

Feature Analysis

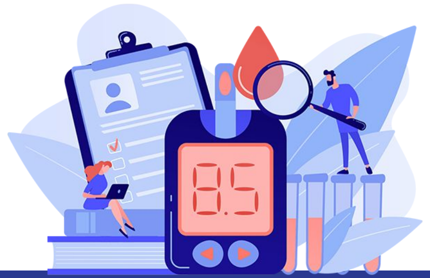
Numpy, Pandas, Seaborn, matplotlib.pyplot, plotly.graph_objs

→

Daiana, Nikita, Priyanshi, Mohammad

Model Selection, Evaluation, & Optimization

KNN, Random Forest, Logistic Regression, DNN, XGBoost, SVC



The Dataset

- Data for this project is taken from 2022 survey data off the CDCs website from their Behavioral Risk Factor Surveillance System (BRFSS) sector.
- The data contains information about U.S. residents health-related risk behaviors, chronic health conditions, and use of preventive services
- https://www.cdc.gov/brfss/annual_data/annual_2022.html
- 326 features (columns) and 445,132 records for 2022

Category	Renamed-as	Label/Question	Value	Null/Refused
_STATE	STATE	*-State FIPS Code*	-Integer [1-78]	--
DISPCODE	DISPCODE	*- Final Disposition*	1100 : Completed Interview 1200 : Partial Complete Interview	--
SEXVAR	GENDER	*-Sex of Respondent*	1: MALE 2: FEMALE	--
_INCOMG1	INCOME	*-Income categories* (Computed income categories)	Integer [1-7]	9: Don't Know/refused
HEIGHT3	HEIGHT	*-About how tall are you without shoes?* (Height in Feet and Inches)	200 - 711 : ft/inches 9061 - 9998 : m/cm	7777 & 9999 : Don't Know/refused BLANK
WTKG3	WEIGHT	*-Computed Weight in Kilograms* (Reported in kilograms)	FLOAT [2300 - 29500]	BLANK
_BMI5CAT	BMI	*-Computed body mass index categories* (Four-categories of BMI)	1: Underweight 2 : Normal Weight 3: Over Weight 4: Obese	BLANK
_RACE1	RACE	*-Computed Race-Ethnicity grouping* (Race/ethnicity categories)	1: White 2: Black 3: Indian/ Alaskan Native 4: Asian 5: Hawaiian/Pacific Islander 7: Multiracial 8: Hispanic	9: Don't Know/refused BLANK

Data Collection & Transformation

	_STATE	SEXVAR	_INCOMG1	HEIGHT3	WTKG3	_BMI5CAT	_RACE1	_AGEG5YR	DIABETE4	PREDIAB2	...
0	1.0	2.0	9.0	9999.0	NaN	NaN	1.0	13.0	1.0	NaN	...
1	1.0	2.0	3.0	503.0	6804.0	3.0	1.0	13.0	3.0	NaN	...
2	1.0	2.0	6.0	502.0	6350.0	3.0	1.0	8.0	3.0	NaN	...
3	1.0	2.									
4	1.0	2.									
5	1.0	1.									
6	1.0	2.									
7	1.0	2.									
8	1.0	2.									
9	1.0	2.									

(445132 rows x 33 columns)

	_STATE	SEXVAR	_INCOMG1	HEIGHT3	WTKG3	_BMI5CAT	_RACE1	_AGEG5YR	DIABETE4	PREDIAB2	...
0	1.0	FEMALE	NaN	9999.0	NaN	NaN	White	13.0	1.0	NaN	...
1	1.0	FEMALE	3.0	503.0	6804.0	Over_Weight	White	13.0	0.0	NaN	...
2	1.0	FEMALE	6.0	502.0	6350.0	Over_Weight	White	8.0	0.0	NaN	...
3	1.0	FEMALE	NaN								
4	1.0	FEMALE	3.0								
5	1.0	MALE	NaN								
6	1.0	FEMALE	5.0								
7	1.0	FEMALE	5.0								
8	1.0	FEMALE	5.0								
9	1.0	FEMALE	5.0								

Categorical values were converted to categories

```
PREDIAB2 : [nan  0.  1.  2.]
DIABTYPE : [nan  1.  2.]
_TOTINDA : [ 0.  1. nan]
PERSDOC3 : [ 1.  2. nan  0.]
CHECKUP1 : [ 1.  0. nan  2.  3.  4.]
PDIABTS1 : [nan  2.  1.  0.  3.  6.  5.  4.]
INSULIN1 : [nan  0.  1.]
EYEEXAM1 : [nan  3.  1.  2.  4.  0.]
DIABEYE1 : [nan  4.  1.  2.  3.  0.]
DIABEDU1 : [nan  0.  6.  3.  5.  4.  1.  2.]
FEETSORE : [nan  0.  1.]
CVDINFR4 : [ 0.  1. nan]
CVDCRHD4 : [ 0.  1. nan]
CVDSTRK3 : [ 0.  1. nan]
HAVARTH4 : [ 0.  1. nan]
DIFFWALK : [ 0.  1. nan]
```

'Don't know/Not sure' or 'Refused' categories were converted to NaN values

All the "No/Never" categories were converted to 0

Data Collection & Transformation

_BMI5CAT	_RACE1	_AGEG5YR	DIABETE4	...	DIABEDU1	FEETSORE	CVDINFR4	CVDCRHD4	CVDSTRK3	HAVARTH4	_SMOKER3	DIFFWALK	_EDUCAG	HEIGHT
NaN	White	13.0	1.0	...	NaN	NaN								
Over_Weight	White	13.0	0.0	...	NaN	NaN								
					NaN	NaN								
					NaN	NaN								
					NaN	NaN								
					...	...								
					NaN	NaN								
					NaN	NaN								
					NaN	NaN								
					NaN	NaN	1.0	0.0	0.0	0.0	Never	0.0	College_N	72.0
					NaN									

(353271 rows × 34 columns)

DIABETE4	799
PREDIAB2	234587
PHYSHLTH	8139
MENTHLTH	6742
DIABTYPE	343296
_TOTINDA	802
SLEPTIM1	4027
PRIMINSR	12334
PERSDOC3	3116
CHECKUP1	4289
PDIABTS1	241504
INSULIN1	342543
CHKHEMO3	343074
EYEEXAM1	342727
DIABEYE1	343884
DIABEDU1	343037
FEETSORE	342536

Had a lot of NaN values: Questions only relevant to diabetic patients

ID	STATE	GENDER	INCOME	WEIGHT	BMI	RACE	AGE	DIABETES	PHYSHLTH	...	PERSONAL_DOC	CHECKUP1	HRT_ATTACK
1	1.0	FEMALE	3.0	6804.0	Over_Weight	White	13.0	0.0	0.0	...	2.0	0.0	0.0
2	1.0	FEMALE	6.0	6350.0	Over_Weight	White	8.0	0.0	2.0	...	1.0	1.0	0.0
4	1.0	FEMALE	3.0	5398.0	Normal_Weight	White	5.0	0.0	2.0	...	2.0	1.0	0.0
6	1.0	FEMALE	5.0	6260.0	Normal_Weight	Black	13.0	0.0	0.0	...	1.0	1.0	0.0
7	1.0	FEMALE	5.0	7348.0	Over_Weight	White	13.0	0.0	0.0	...	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
445126	78.0	MALE	5.0	10433.0	Obese	White	3.0	0.0	0.0	...	0.0	2.0	0.0
445127	78.0	FEMALE	1.0	6985.0	Over_Weight	Black	1.0	0.0	0.0	...	0.0	2.0	0.0
445128	78.0	FEMALE	5.0	8301.0	Over_Weight	Black	7.0	0.0	2.0	...	2.0	1.0	0.0
445130	78.0	MALE	5.0	10886.0	Obese	Black	11.0	0.0	0.0	...	2.0	1.0	1.0
445131	78.0	MALE	2.0	6350.0	Normal_Weight	Black	5.0	0.0	0.0	...	0.0	0.0	0.0

General Information Dataframe (246050 rows × 24 columns): To predict if someone has diabetes or not

ID	GENDER	AGE	BMI	DIABETES	DIABTYPE	INSULIN_Y/N	A-one-C_test	EYEEXAM1	DIABEYE1	DIAB_MNGMT	FEETSORE	PERSONAL_DOC
55982	FEMALE	13.0	Over_Weight	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0
55988	MALE	13.0	Over_Weight	1.0	2.0	0.0	2.0	1.0	1.0	6.0	0.0	1.0
55992	MALE	11.0	Over_Weight	1.0	2.0	0.0	2.0	2.0	2.0	0.0	0.0	1.0
55995	MALE	11.0	Over_Weight	1.0	2.0	0.0	1.0	3.0	3.0	3.0	0.0	1.0
56001	MALE	12.0	Obese	1.0	2.0	1.0	4.0	2.0	2.0	0.0	0.0	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...
445060	MALE	11.0	Normal_Weight	1.0	2.0	0.0	4.0	2.0	2.0	0.0	0.0	2.0
445080	MALE	10.0	Over_Weight	1.0	2.0	0.0	2.0	2.0	2.0	6.0	1.0	1.0
445097	MALE	13.0	Normal_Weight	1.0	1.0	1.0	4.0	2.0	2.0	0.0	0.0	2.0
445112	MALE	1.0	Underweight	1.0	2.0	1.0	2.0	0.0	0.0	0.0	0.0	2.0
445124	MALE	10.0	Over_Weight	1.0	2.0	0.0	1.0	0.0	2.0	0.0	0.0	1.0

Diabetic Dataframe (9975 rows × 17 columns): To predict the type of diabetes

Database Setup & Access

Creation of diabetes-database
hostname:
diabetes-dataset.cwpas6tssjkb.us-east-1.rds
.amazonaws.com

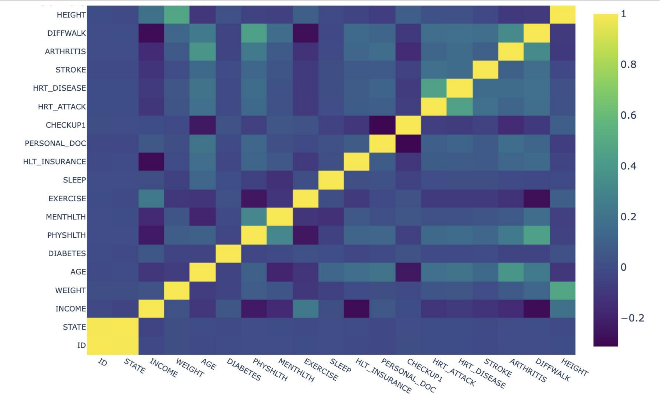
Link diabetes-database to pgAdmin & create user accounts

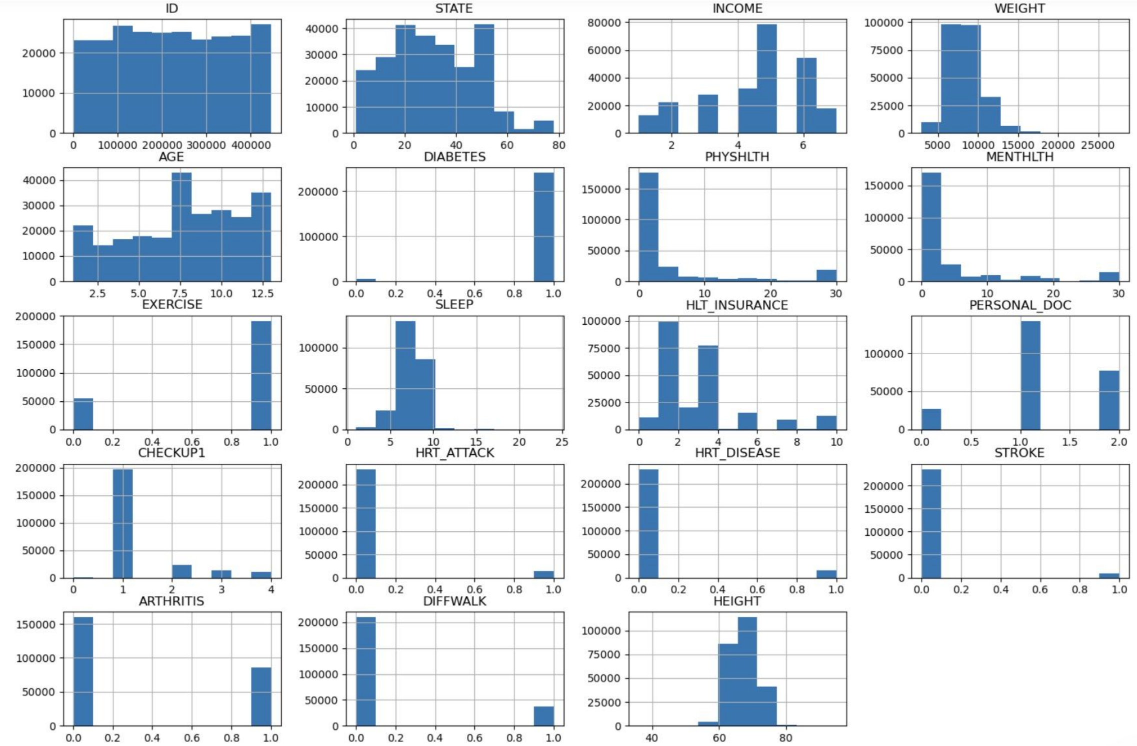
Psycopg2, Sqlalchemy

```
# example query to grab all of the columns
sql_query = "SELECT * FROM general_info"
df = pd.read_sql_query(sql_query, conn)
df.head()
```

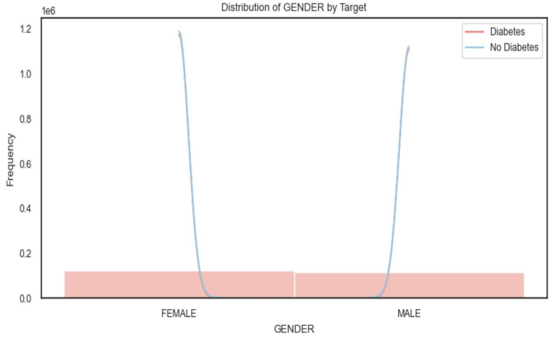
Exploratory Data Analysis

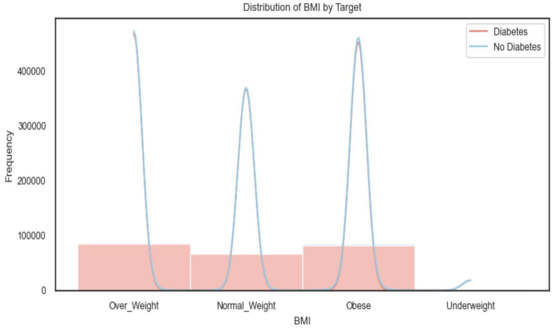
- Univariate Analysis
- Bivariate Analysis
- Correlation Matrix

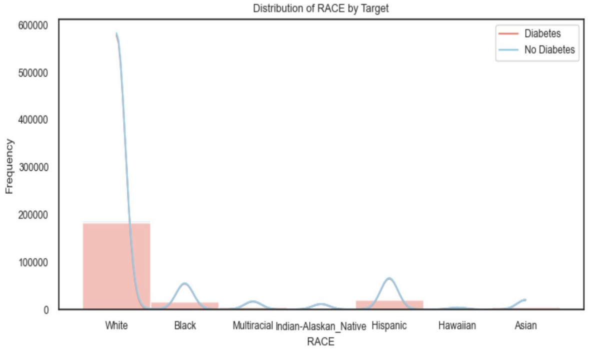


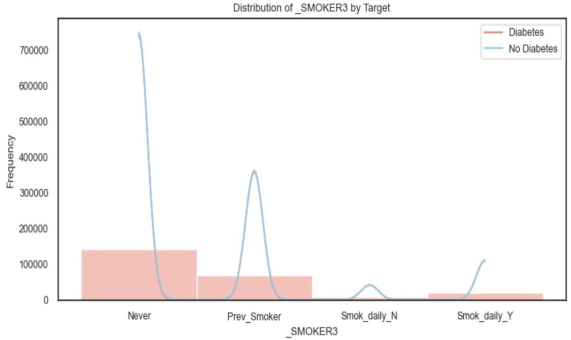


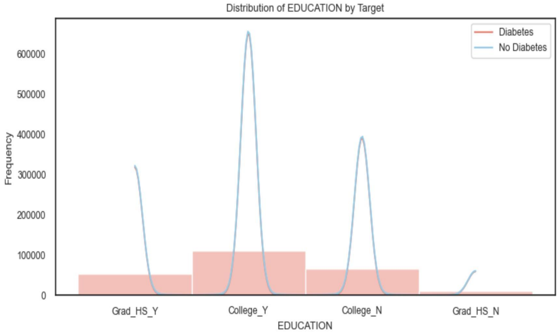
Exploratory Data Analysis

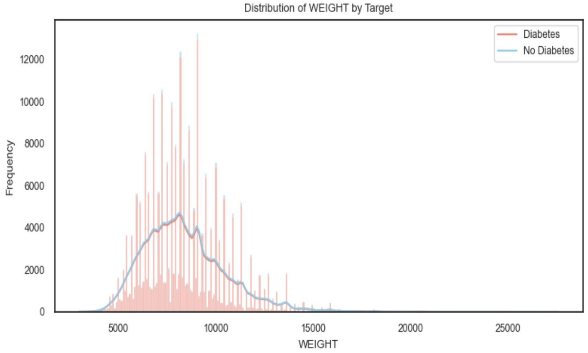




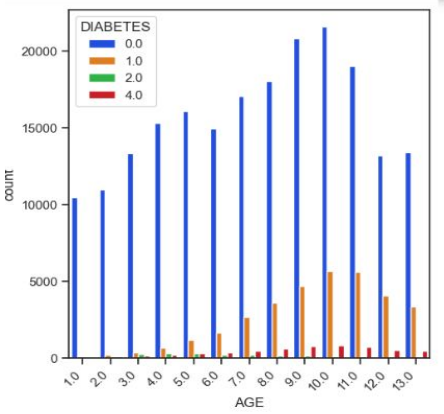


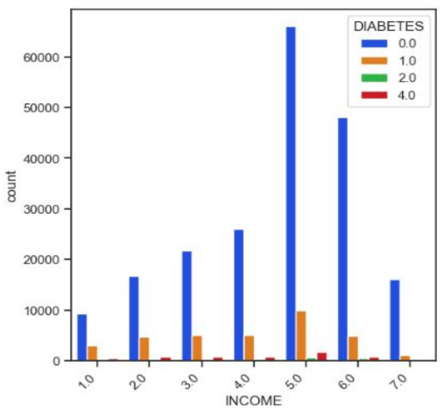


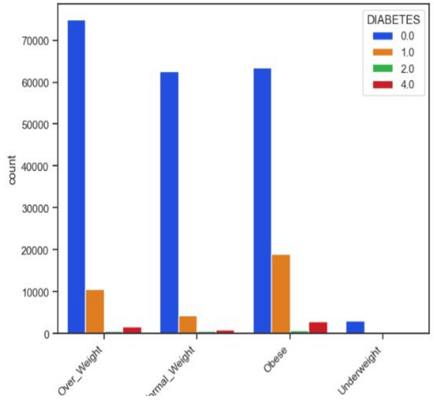


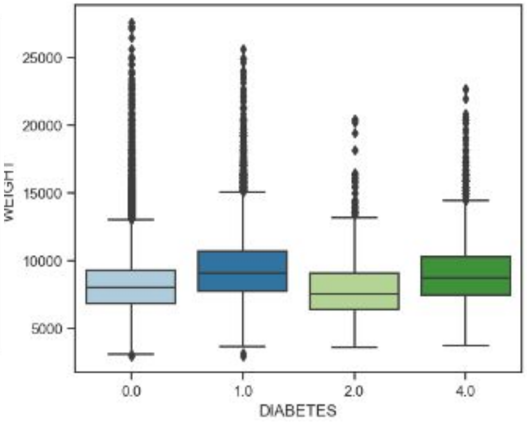


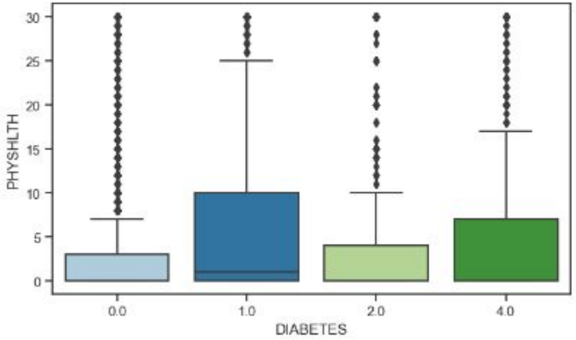
Exploratory Data Analysis

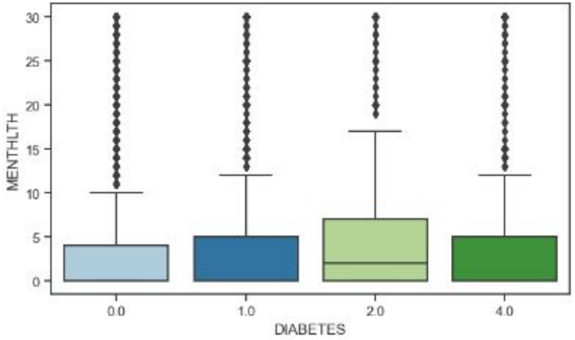


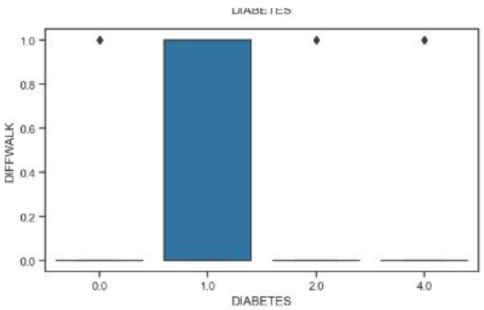












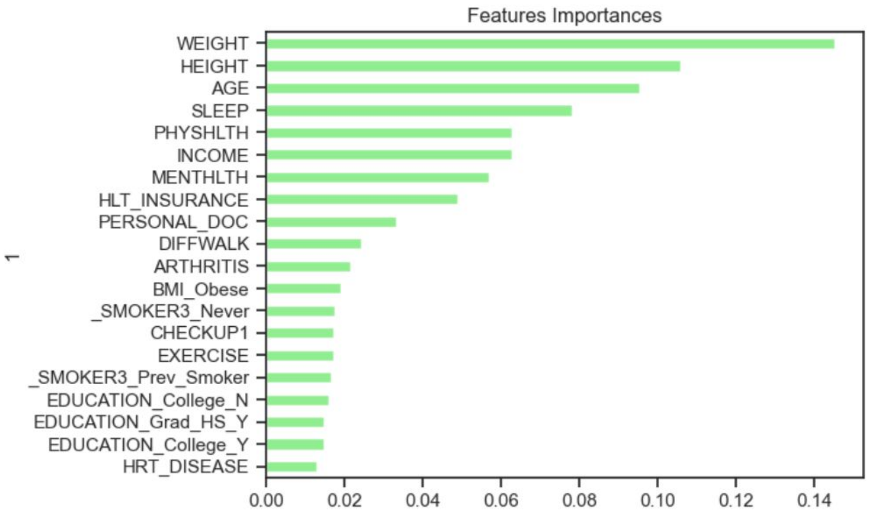
Feature Analysis

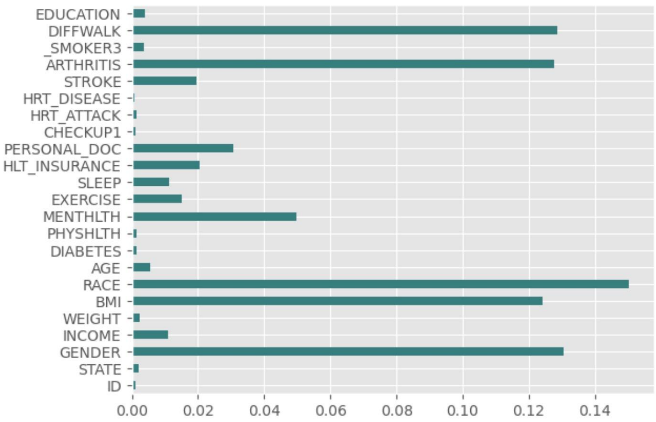
LIBRARIES

- # Statistics
- Pandas, numpy, scipy.stats, category_encoders, sklearn.feature_selection
- # Plots
- Seaborn, matplotlib.pyplot, plotly.

FEATURE SELECTION

- Mutual Information
- Chi-Square Test (categorical variable)
- Correlation Coefficient





Model Selection

- Binary classification: has diabetes, no diabetes
  - K-Nearest Neighbours (k=2)
  - Random Forest
  - Deep Neural Network
  - Logistic Regression
- Binary classification: type 1, type 2
  - Support Vector Machine
  - XGBoost
  - Random Forest

Libraries used: Numpy, pandas, sklearn, tensorflow, train_test_split

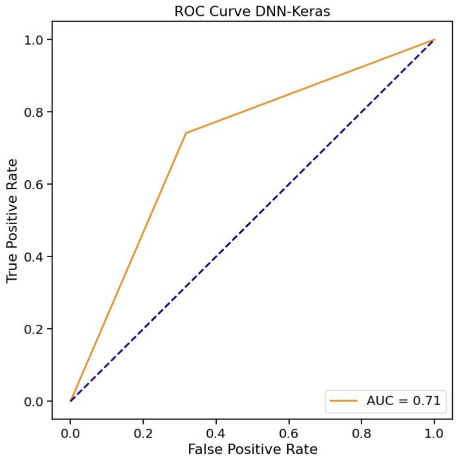
Model Building & Evaluation

- Evaluation metrics:
  - Confusion matrix
  - Accuracy
  - F1 score
  - Precision
  - Recall
- Resampling techniques:
  - Random Under Sampling
  - RandomOversampler

Model Results - Dataset 1

GENDER	INCOME	WEIGHT	BMI	RACE	AGE	DIABETES
FEMALE	3.0	6804.0	Over_Weight	White	13.0	0.0
FEMALE	6.0	6350.0	Over_Weight	White	8.0	0.0
FEMALE	3.0	5398.0	Normal_Weight	White	5.0	0.0
FEMALE	5.0	6260.0	Normal_Weight	Black	13.0	0.0
FEMALE	5.0	7348.0	Over_Weight	White	13.0	0.0

	Accuracy	Precision	F1 Score	Recall	Model Type	Resample
0	0.611921	0.679167	0.522215	0.424187	KNN	50/50 Split
1	0.699871	0.687136	0.709722	0.733843	Random Forest	50/50 Split
2	0.701783	0.692934	0.708444	0.724665	DNN	50/50 Split
3	0.759709	0.288277	0.284663	0.281139	KNN	Oversampled Data
4	0.813698	0.422642	0.322615	0.260874	Random Forest	Oversampled Data
5	0.679645	0.310654	0.434937	0.724978	DNN	Oversampled Data
6	0.712012	0.700226	0.720223	0.741396	DNN	Keras
7	0.817811	0.360821	0.147173	0.092439	KNN	Reduced Features

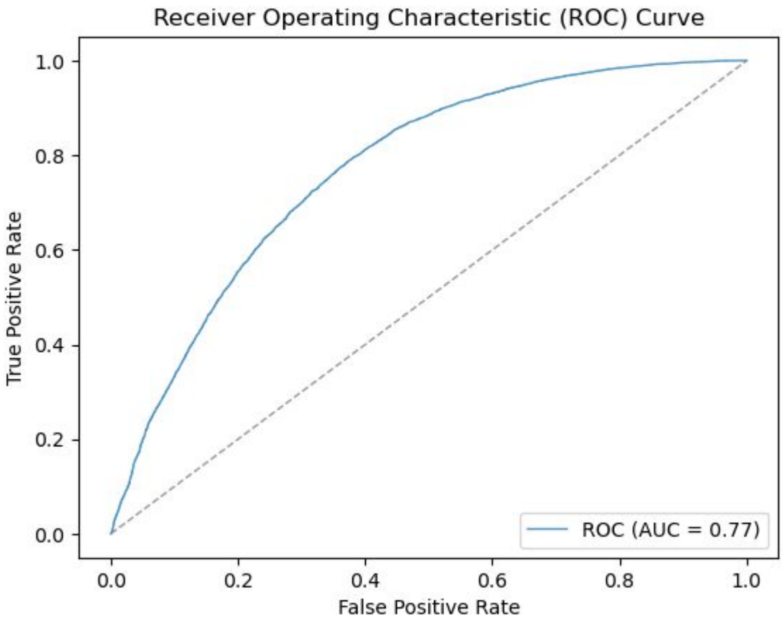



```
Classification Report
============================================================
              precision    recall  f1-score   support

         0.0       0.73      0.68      0.70     10461
         1.0       0.70      0.74      0.72     10460

    accuracy                           0.71     20921
   macro avg       0.71      0.71      0.71     20921
weighted avg       0.71      0.71      0.71     20921
```

Model Results - Dataset 2

	DISPCODE	Diabetes	Smoker	CHD_1	CHD_2	Alcohol	GeneralHealth	MentalHealth	PhysicalHealth	Sex	Age	Education	Income
0	1100.0	1.0	4.0	2.0	2.0	1.0	2.0	88.0	88.0	2.0	13.0	4.0	99.0
1	1100.0	3.0	4.0	2.0	2.0	1.0	1.0	88.0	88.0	2.0	13.0	2.0	5.0
2	1100.0	3.0	4.0	2.0	2.0	1.0	2.0	3.0	2.0	2.0	8.0	4.0	10.0
3	1100.0	3.0	2.0	2.0	2.0	1.0	1.0	88.0	88.0	2.0	14.0	2.0	77.0
4	1100.0	3.0	4.0	2.0	2.0	1.0	4.0	88.0	2.0	2.0	5.0	3.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
445127	1100.0	3.0	4.0	2.0	2.0	9.0	3.0	3.0	88.0	2.0	1.0	2.0	1.0
445128	1100.0	3.0	4.0	2.0	2.0	1.0	1.0	2.0	2.0	2.0	7.0	4.0	7.0
445129	1100.0	3.0	1.0	2.0	2.0	9.0	5.0	30.0	30.0	2.0	10.0	2.0	77.0
445130	1100.0	3.0	4.0	2.0	1.0	1.0	2.0	88.0	88.0	1.0	11.0	3.0	8.0
445131	1100.0	3.0	3.0	2.0	2.0	9.0	2.0	1.0	88.0	1.0	5.0	1.0	4.0

```
              precision    recall  f1-score   support

         0.0       0.69      0.70      0.70      9300
         1.0       0.71      0.70      0.70      9566

    accuracy                           0.70     18866
   macro avg       0.70      0.70      0.70     18866
weighted avg       0.70      0.70      0.70     18866
```



Model Results - Dataset 3- Random Forest

	Predicted 0	Predicted 1
Actual 0	49902	1282
Actual 1	9026	1303

```
Accuracy Score : 0.8324256661193569
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.97      0.91     51184
           1       0.50      0.13      0.20     10329

    accuracy                           0.83     61513
   macro avg       0.68      0.55      0.55     61513
weighted avg       0.79      0.83      0.79     61513
```

	Predicted 0	Predicted 1
Actual 0	46372	4774
Actual 1	714	50244

```
Accuracy Score : 0.9462508814542036
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.91      0.94     51146
           1       0.91      0.99      0.95     50958

    accuracy                           0.95    102104
   macro avg       0.95      0.95      0.95    102104
weighted avg       0.95      0.95      0.95    102104
```

```
Y=df_dummies["DIABETIC"]
Y.value_counts()
```

```
0    204208
1     41842
Name: DIABETIC, dtype: int64
```

Model Results - Dataset 4

Two models provided the best results, XGB and random Forest For category

```
In [119]: xgb_classifier_1 = xgb.XGBClassifier(objective='multi:softmax', num_class=2, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y_1, test_size=0.2, random_state=42)

X_scaler = skl.preprocessing.StandardScaler()

# Fit the scaler
X_scaler.fit(X_train)

# Scale the data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)

# Train the model on the training data
xgb_classifier_1.fit(X_train_scaled, y_train)

# Make predictions on the test data
y_pred = xgb_classifier_1.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(classification_report(y_test, y_pred))

Accuracy: 0.899749373433584
Classification Report:
              precision    recall  f1-score   support

           0       0.42      0.12      0.19       191
           1       0.91      0.98      0.95      1804

    accuracy                           0.90      1995
   macro avg       0.67      0.55      0.57      1995
weighted avg       0.87      0.90      0.87      1995
```

```
In [ ]:

In [146]: xgb_classifier = xgb.XGBClassifier(objective='multi:softmax', num_class=2, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y_1, test_size=0.2, random_state=42)

# Train the model on the training data
xgb_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = xgb_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(classification_report(y_test, y_pred))

Accuracy: 0.9027568922305764
Classification Report:
              precision    recall  f1-score   support

           0       0.47      0.13      0.20       191
           1       0.91      0.98      0.95      1804

    accuracy                           0.90      1995
   macro avg       0.69      0.56      0.58      1995
weighted avg       0.87      0.90      0.88      1995
```

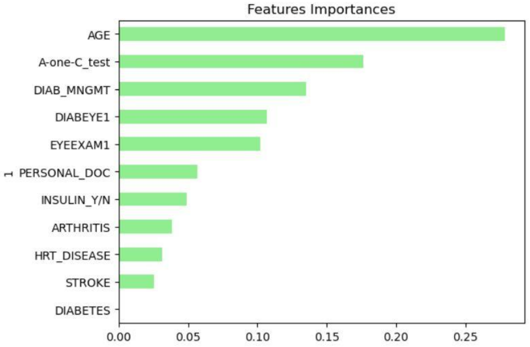
```
In [114]: random_forest_1 = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
random_forest_1.fit(X_train_scaled, y_train)

# Make predictions on the test data
y_pred = random_forest.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(classification_report(y_test, y_pred))

Accuracy: 0.8376102646351243
Classification Report:
              precision    recall  f1-score   support

         1.0       0.10      0.11      0.11       213
         2.0       0.92      0.91      0.91      2281

    accuracy                           0.84      2494
   macro avg       0.51      0.51      0.51      2494
weighted avg       0.85      0.84      0.84      2494

D:\Ubisoft\New folder\envs\dev\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

```
In [77]: from sklearn.ensemble import RandomForestClassifier

In [78]: random_forest = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
random_forest.fit(X_train, y_train)

# Make predictions on the test data
y_pred = random_forest.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(classification_report(y_test, y_pred))

Accuracy: 0.8982456140350877
Classification Report:
              precision    recall  f1-score   support

         1.0       0.41      0.14      0.20       191
         2.0       0.91      0.98      0.95      1804

    accuracy                           0.90      1995
   macro avg       0.66      0.56      0.57      1995
weighted avg       0.87      0.90      0.87      1995
```

Model Results - Dataset 4

Two models provided the best results, XGB and random Forest for category

```
In [134]: importances_df = pd.DataFrame(sorted(zip(random_forest.feature_importances_, X.columns), reverse=True))
          importances_df.set_index(importances_df[1], inplace=True)
          importances_df.drop(columns=1, inplace=True)
          importances_df.rename(columns={0: 'Feature Importances'}, inplace=True)
          importances_sorted = importances_df.sort_values(by='Feature Importances')
          importances_sorted.plot(kind='barh', color='lightgreen', title= 'Features Importances', legend=False)

Out[134]: <Axes: title={'center': 'Features Importances'}, ylabel='1'>
```




```
In [139]: random_forest_2 = RandomForestClassifier(n_estimators=100, random_state=42)

          # Train the model on the training data
          random_forest_2.fit(X_train, y_train)

          # Make predictions on the test data
          y_pred = random_forest_2.predict(X_test)

          # Evaluate the model
          accuracy = accuracy_score(y_test, y_pred)
          print(f"Accuracy: {accuracy}")
          print("Classification Report:")
          print(classification_report(y_test, y_pred))

Accuracy: 0.8957393483709273
Classification Report:
              precision    recall  f1-score   support

         1.0       0.39      0.15      0.22       191
         2.0       0.92      0.97      0.94      1804

    accuracy                           0.90      1995
   macro avg       0.65      0.56      0.58      1995
weighted avg       0.86      0.90      0.87      1995
```

Summary of Results

	Diabetes yes/no prediction			Diabetes Type Prediction
	Dataset 1	**Dataset 2**	**Dataset 3**	**Dataset 4**
Accuracy	0.712012	0.70083	0.9462	0.8957
F1 Score	0.700226	0.70183	0.94	0.94
Precision	0.720223	0.71076	0.91	0.92
Recall	0.741396	0.70062	0.99	0.97
Model	-Hyperparameter tuning -Undersampled dataset -Neural Network -Subset of the data	- Equal Dataset - Over and Under Dataset	-Random Forest -Random Oversampling -Entire dataset (34 features)	-Scaling the dataset -Importance dataset formation and random forest model

Conclusion

- Preparing the data is very important for achieving good results
- More features lead to better model performance
- Having a balanced dataset is important to have higher recall and precision for the minority class
- Some features that may not look significant may play a big role in classification
- Future improvements: adding more data for the diabetic patients including sugar level, cholesterol, etc

Questions?

Thank you!
Hope to see you all soon!!