Diego Izaguirre

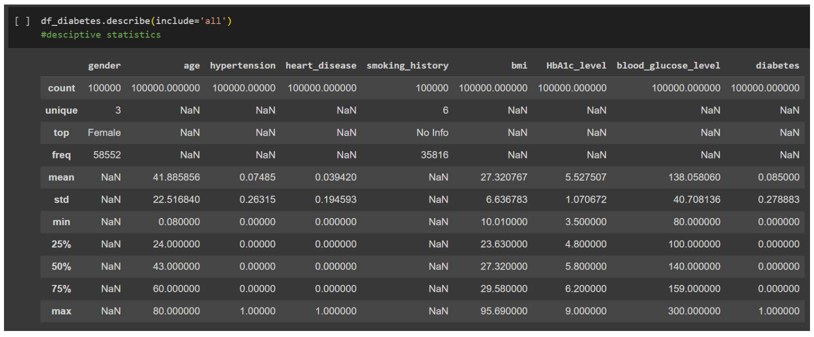
6/6/2023

Final Report

Data Exploration

The Diabetes prediction dataset was used in two supervised and one unsupervised machine learning algorithms in order to predict whether or not a certain individual may have diabetes. The dataset is a collection of medical and demographic data from patients, along with their diabetes status (positive or negative). There are 9 features with 100,000 observations. Age is a numerical feature that indicates the age of the person. Hypertension is a categorical feature that indicates a value of 1 means positive for hypertension and a 0 means negative. Heart_disease is a categorical feature; A value of 1 indicates positive and a 0 indicates negative. Bmi is numerical that indicates the body mass index of the individual. hA1c is a numerical variable that measures the average blood glucose level based on the presence of glycated hemoglobin in the blood. Blood_glucose_level is a numerical feature indicating the level of glucose in the blood of an individual. Gender is a categorical feature which indicates whether an individual is male, female, or other. Smoking_history is a categorical feature that indicates whether an individual is current, never, former, not current, or if there was no info. Then the target variable, diabetes, indicates whether an individual is positive for diabetes or not; a 1 indicates positive and a 0 indicates negative.

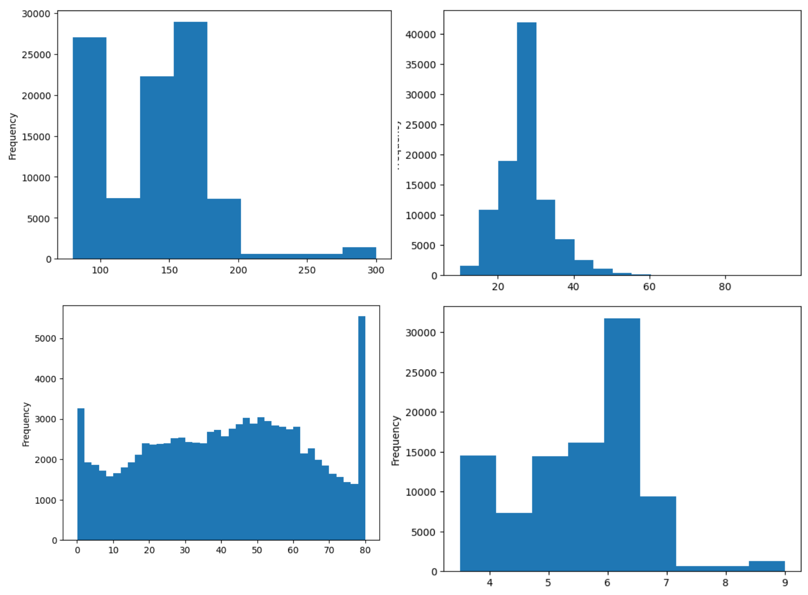
Some descriptive statistics and data visualizations were created in order to better explore the data. On the right are some



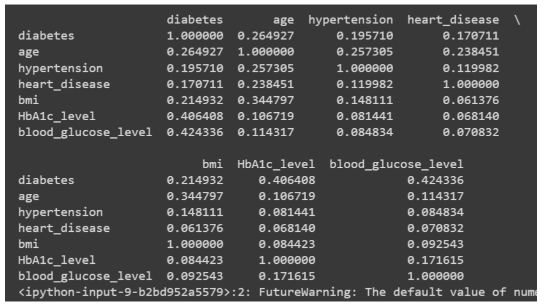
```
[ ] df_diabetes.describe(include='all')
    #desciptive statistics
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
count	100000	100000.000000	100000.00000	100000.000000	100000	100000.000000	100000.000000	100000.000000	100000.000000
unique	3	NaN	NaN	NaN	6	NaN	NaN	NaN	NaN
top	Female	NaN	NaN	NaN	No Info	NaN	NaN	NaN	NaN
freq	58552	NaN	NaN	NaN	35816	NaN	NaN	NaN	NaN
mean	NaN	41.885856	0.07485	0.039420	NaN	27.320767	5.527507	138.058060	0.085000
std	NaN	22.516840	0.26315	0.194593	NaN	6.636783	1.070672	40.708136	0.278883
min	NaN	0.080000	0.00000	0.000000	NaN	10.010000	3.500000	80.000000	0.000000
25%	NaN	24.000000	0.00000	0.000000	NaN	23.630000	4.800000	100.000000	0.000000
50%	NaN	43.000000	0.00000	0.000000	NaN	27.320000	5.800000	140.000000	0.000000
75%	NaN	60.000000	0.00000	0.000000	NaN	29.580000	6.200000	159.000000	0.000000
max	NaN	80.000000	1.00000	1.000000	NaN	95.690000	9.000000	300.000000	1.000000

descriptives for the features in the dataset. Though not particularly helpful for the categorical variables, it helps to understand the distributions of the numerical variables. These distributions were also visualized using histograms. The figures on the right are the visualized distributions for all the numerical variables. From left to right they are: blood glucose, BMI, age, and hbA1c. As we can see the BMI distribution is right skewed as are blood glucose and hba1c. This is likely due to outliers in the dataset. The distribution of age is interesting as it indicates that there are a lot of elderly and babies included in the dataset.



A correlation matrix was also made on the features in order to see which features may be correlated with each other. The figure on the right is the correlation matrix, looking at the first column under diabetes we can see how all the features are correlated with the target variable. Unsurprisingly, the highest correlated variables are hbA1c and blood glucose. Age is also correlated along with bmi indicating someone's weight and age will impact their probability of getting diabetes.

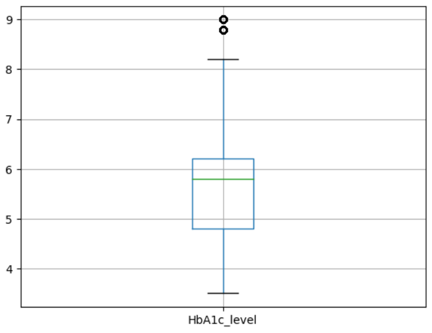


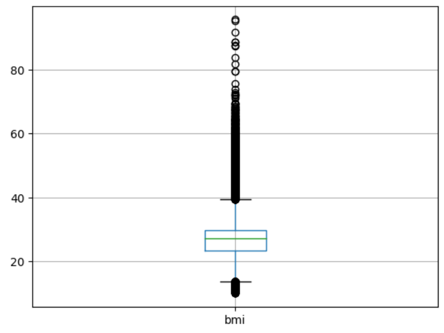
```
                     diabetes       age  hypertension  heart_disease  \
diabetes             1.000000  0.264927      0.195710       0.170711
age                  0.264927  1.000000      0.257305       0.238451
hypertension         0.195710  0.257305      1.000000       0.119982
heart_disease        0.170711  0.238451      0.119982       1.000000
bmi                  0.214932  0.344797      0.148111       0.061376
HbA1c_level          0.406408  0.106719      0.081441       0.068140
blood_glucose_level  0.424336  0.114317      0.084834       0.070832

                          bmi  HbA1c_level  blood_glucose_level
diabetes             0.214932     0.406408             0.424336
age                  0.344797     0.106719             0.114317
hypertension         0.148111     0.081441             0.084834
heart_disease        0.061376     0.068140             0.070832
bmi                  1.000000     0.084423             0.092543
HbA1c_level          0.084423     1.000000             0.171615
blood_glucose_level  0.092543     0.171615             1.000000
<ipython-input-9-b2bd952a5579>:2: FutureWarning: The default value of nume
```

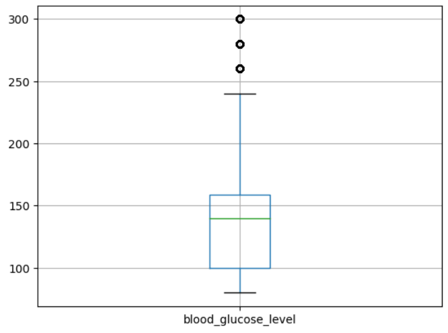
Cleaning and Preprocessing

In terms of dimensionality and numerosity there were no missing values. There was also no need to do a dimensionality reduction because there were enough features for the number of observations in the dataset. In fact, since there were so many observations, I did decide to do a numerosity reduction by removing any duplicate observations from the dataset. After doing so the new number of observations is 96146. Additionally, I noticed a typo in the dataset that misspelled “never” and as such added an additional dummy variable called “ever”. The observation containing the typo was removed.

The only features with outliers are BMI, hbA1c level, and blood glucose level. The visualization of these outliers is shown in the boxplots on the right. BMI has a large number of outliers compared to the other two features. After research these particular features the values of these outliers are possible, although dangerous, to have. Therefore I opted to keep them in the dataset. But because there is such a large amount of outliers, BMI was sorted into bins and then converted into dummy variables. Low BMI has the range 10-17, Normal BMI has the range 18-28, and high BMI has the range 29-95. Dummy variables also had to be made for the categorical features with qualitative data. Hypertension and heart disease were already set up as binary variables. So gender, smoking history, and BMI were sorted into dummy variables.







I will be using KNN in this project and since these algorithms are based on distance, the data will need to be normalized because if the data points are two far away from each other because of differing scales, the algorithm will not yield accurate results. Z-score normalization will be used on account of many of the feature distributions being skewed due to outliers. I will also be using random forest but the data will not be normalized for that algorithm. An 80/20 split was used to split training and testing respectively. No validation set was used. For the unsupervised machine learning model an 80/20 split was also used in order to attempt to replicate the initial results on unseen data. If they are successfully replicated then the results can be validated.

Supervised Model- K Nearest Neighbor

K-Nearest Neighbor (KNN) is a supervised machine learning algorithm that can be used for both classification and regression tasks. It works by finding the k nearest neighbors of a given data point and then assigning the label that is most common among those neighbors.

The hyperparameters that need to be chosen for this model are the value of K, which is the number of neighbors around the given datapoint, and the distance metric, which will designate how the distance between the points is measured.

In my model the value of K was set to 30 because after that there is no more improvement in the model. So the following chart tracks performance of different distance measures with K=30 for the testing and training set.

	Accuracy	Precision	Recall
Manhattan	Testing:`0.998` Training:`0.957`	Testing:`0.955` Training:`0.994`	Testing:`0.498` Training:`0.509`
Cosine	Testing:`0.957` Training:`0.958`	Testing:`0.987` Training:`0.985`	Testing:`0.520` Training:`0.528`
Euclidean	Testing:`0.956` Training:`0.988`	Testing:`0.991` Training:`0.957`	Testing:`0.513` Training:`0.519`

Something to note when looking at the performance metrics is that we really only want to look at recall. Since we have a class imbalance issue in the target variable, recall is a more accurate measure of performance than accuracy or precision. When there is a class imbalance, the model may learn to classify everything towards the dominant class. This means that the accuracy metric will be biased towards the dominant class and may not accurately reflect the performance of the model on the minority class. Additionally because this is a medical dataset it is important to correctly diagnose those who actually have the disease, rather than misdiagnose those that do not.

So looking at the performance the best hyperparameters for this model are K=30 using cosine distance. This yields a training recall of 0.528 and a testing of 0.520. This means that out of all the actual positives 52% of them were classified as positives by the algorithm. This is not very good performance as about half of those with diabetes would be classified as not having it. This poor performance is expected due in part to the class imbalance issue and the large number of observations which can lead to poor classification of the minority class, which in this case is positive cases of diabetes. Though the model does not show any overfitting/underfitting since the testing and training performances are nearly identical.

Supervised Model–Random Forest

Random forest is a supervised machine learning algorithm that uses multiple decision trees to make predictions. It is a powerful algorithm that can be used for a variety of tasks, including classification and regression. Each tree consists of a data sample drawn from a training set with replacement, called the bootstrap. A label is then assigned by the class selected my the most trees

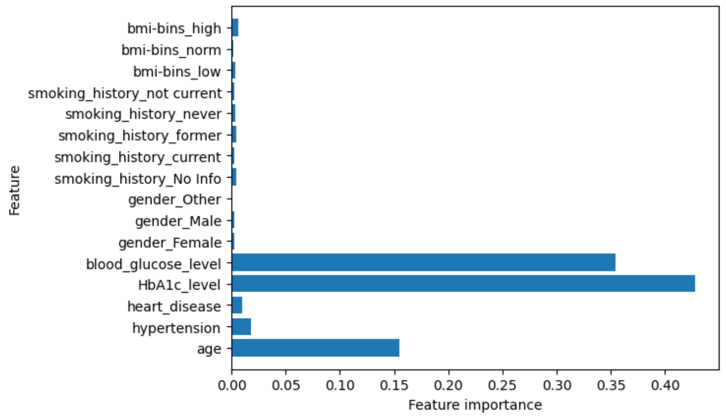
The hyperparameters that need to be chosen are n_estimators which is the number of trees in the model, max_depth which is the maximum depth of the trees, min_samples_split is how many samples a node needs before it can split, min_samples_leaf is the minimum number of samples needed to be a leaf node.

In my model n_estimators was set to 20 after starting with 10 because after that there is no more improvement in the model. The rest of the hyperparameters were kept at the default value since any increase in these parameters saw a big decrease in the recall which was not ideal for this particular dataset. The following is the performance output for the following hyperparameters: n_estimators = 20, max_depth = None, min_samples_split = 2, min_samples_leaf = 1

	Accuracy	Precision	Recall
Training	`0.991`	`0.975`	`0.918`
Testing	`0.966`	`0.859`	`0.710`

The performance on the training is phenomenal. The recall has a value of 0.918 which indicates that 92% of actual positives were classified as positive. Therefore most people are correctly diagnosed with diabetes with this model. The testing underperformed a little bit with a recall of 0.710. This means that 71% of actual positives were classified as positive. While this is still a majority of correct classifications, there is a substantial difference between the training and testing. This may indicate an overfitting issue because of the class imbalance. The training and testing sets may have different ratios of negative and positive cases of diabetes. A solution to this may be to randomly sample observations so that there are an equal proportion of negatives and positives in both the training and testing sets.

The figure on the right is a graph that charts the importance of the random forest model. The three most important features for predicting diabetes are age, hbA1c level, and blood_glucose_level. And comparatively, the rest of the features hold very little sway in whether or not someone might get diabetes. Because some of these features are so unimportant it may open up the opportunity to simplify the model by removing some redundant variables.

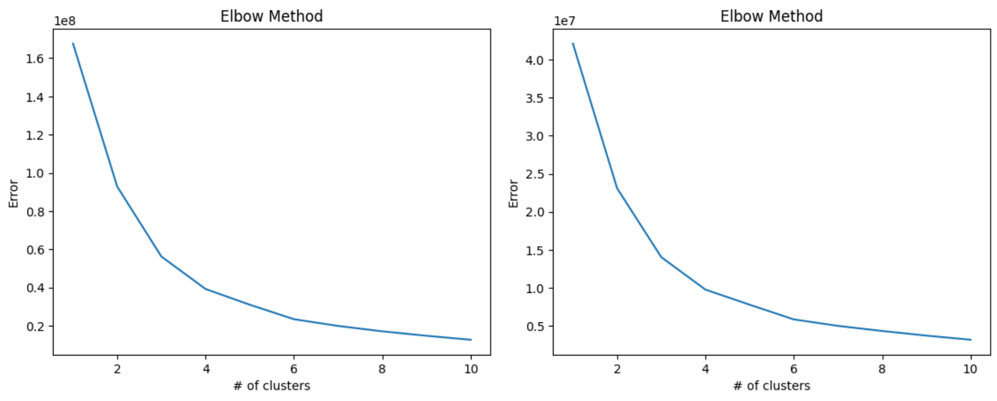


Unsupervised Model-K means

K-means is a clustering algorithm that groups data points into a set of k clusters. The algorithm works by iteratively assigning each data point to the cluster with the nearest mean, and then recalculating the cluster means. This process is repeated until the cluster assignments no longer change.

The hyperparameter for the K-means algorithm is the value of K which designates how many clusters as well as the number of centroids to initially generate. Since I am looking to classify someone as positive or negative for diabetes I choose K=2 initially. After running the WCSS plot vs the number of clusters and using the elbow method it seems that the better number of clusters would be 4. Since this algorithm uses unlabeled data the typical performance metrics cannot be used. Instead I opted to use the training set to replicate the results.

In order to verify this I ran another WCSS plot on the testing set. From left to right are the training and testing plots respectively. They are pretty much nearly identical so I can conclude that 4 is the ideal number of clusters for this dataset. What I infer from this is that it may be better to use this model to classify how likely someone is to get diabetes rather than just say if they have it or not. So, for example, this algorithm might be better suited to classify someone as being no risk, some risk, high risk, very high risk.



Conclusion

In conclusion, random forest was the best performing supervised learning model. KNN did not perform well on this dataset, due in part because of the large number of observations but also the class imbalance. The K means algorithm also grouped the data into 4 clustered instead of ideally just 2. This although not what was expected, discovers a new pattern in the data that I had not thought of before. Some improvement I would make to these models would be to include a validation set for parameter tuning and correcting the class imbalance in the target variable in order to hopefully get better performance from the random forest model.