Diabetes: Data Analysis and Manipulation

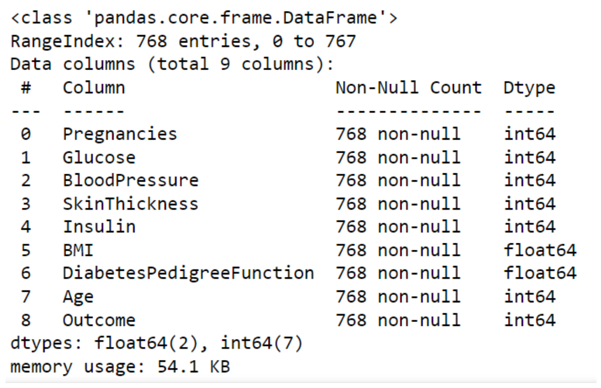
Justin Park

Frame the Problem and Look at the Big Picture

Problem: Patients with diabetes usually do not get tested until they have random symptoms and see a doctor. Being able to predict a patient's risk of having diabetes based on certain attributes would allow them to save time and possibly diagnose a diabetes case early on. This model will allow the user to predict someone's risk to diabetes based on 8 attributes about the person.

Get the Data

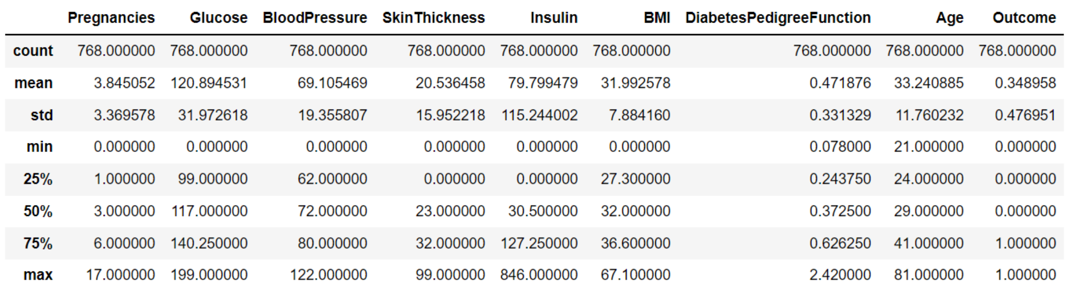
Dataset: https://www.kaggle.com/datasets/whenamancodes/predict-diabities



```
<class 'pandas.core.frame.DataFrame'>
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
```

Figure 1

I got the data from Kaggle. Figure 1 shows the 8 attributes (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age) and the 1 label (Outcome) that we will be using in this project. The info shows that all the data types are either int or float and all values are non-null. There are only 768 instances in the dataset which makes me think that the accuracy of the trained model will not be super high.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Figure 2

Figure 2 shows the ranges, mean, and std of the attributes and label. One thing that I noticed is that although none of those values are null, some of them do not make sense. The minimum value of blood pressure, skin thickness, and BMI are zero.

Explore the Data to Gain Insights

Figure 3 shows the distribution of the attributes and labels. As mentioned earlier there are some infeasible values that are shown in the bar graphs. Also, there are about twice as many instances in the dataset that have diabetes vs the number that do not.

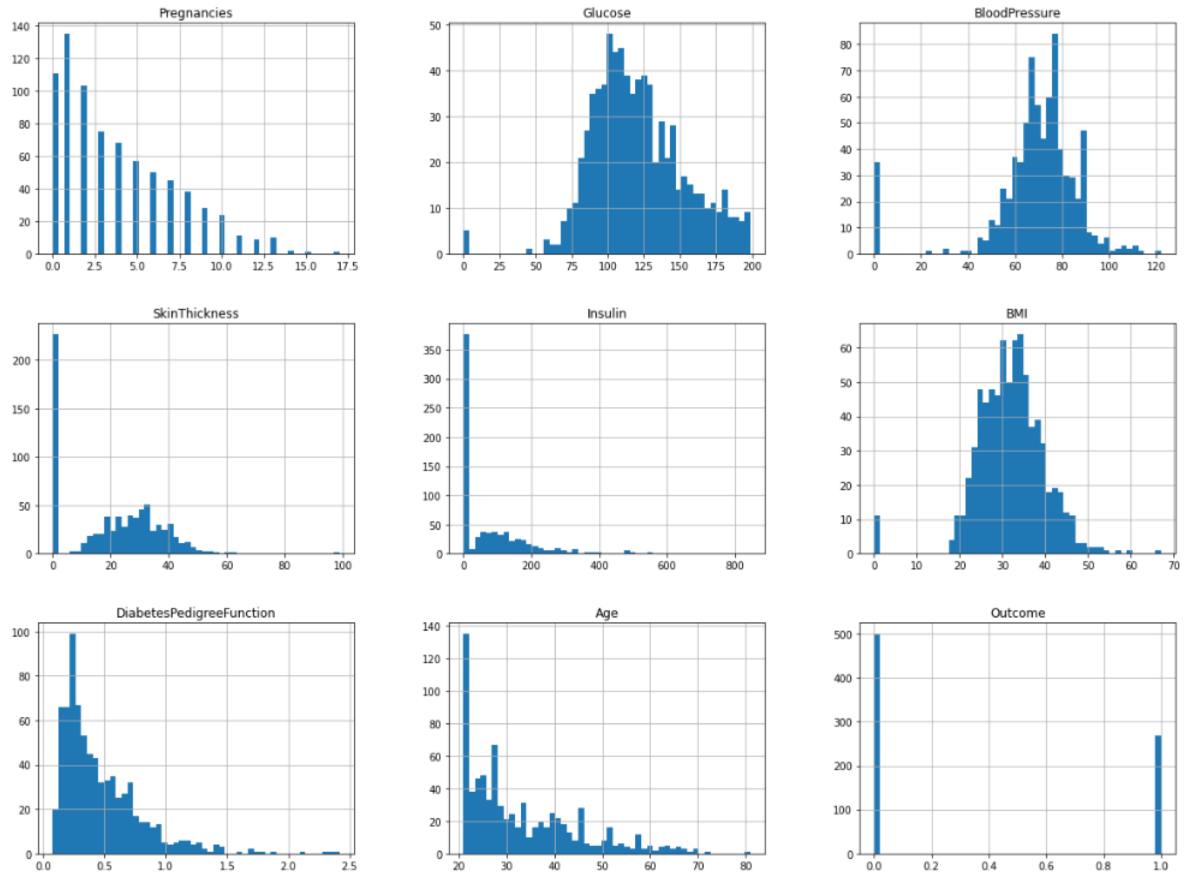


Figure 3

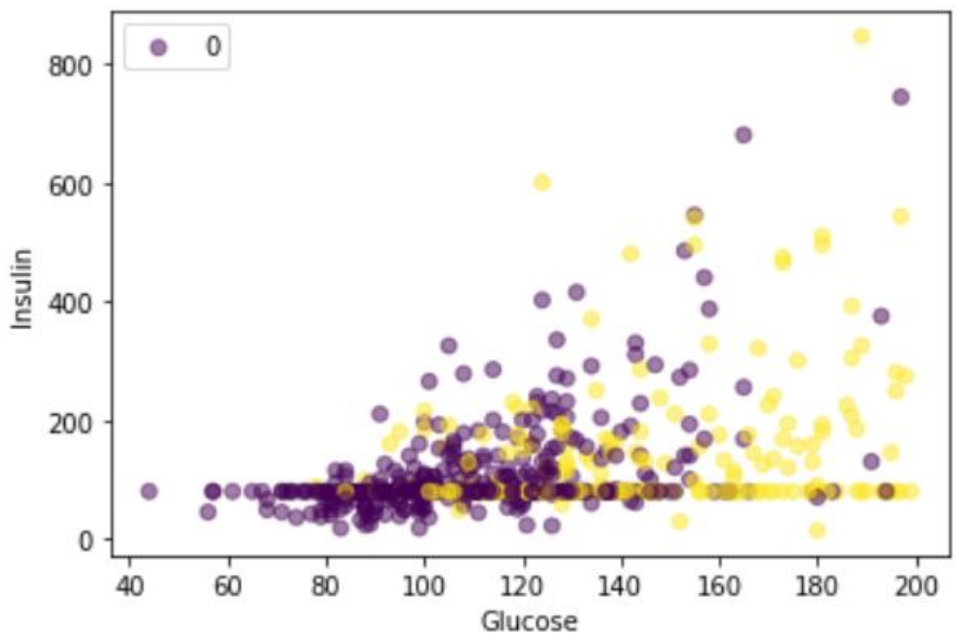


Figure 4

Figure 4 shows the correlation between insulin and glucose. The yellow dots are the instances where the patient has diabetes, and the purple dots are the patients without diabetes. It looks like the people with higher glucose levels tend to have diabetes more often than those with lower glucose levels.

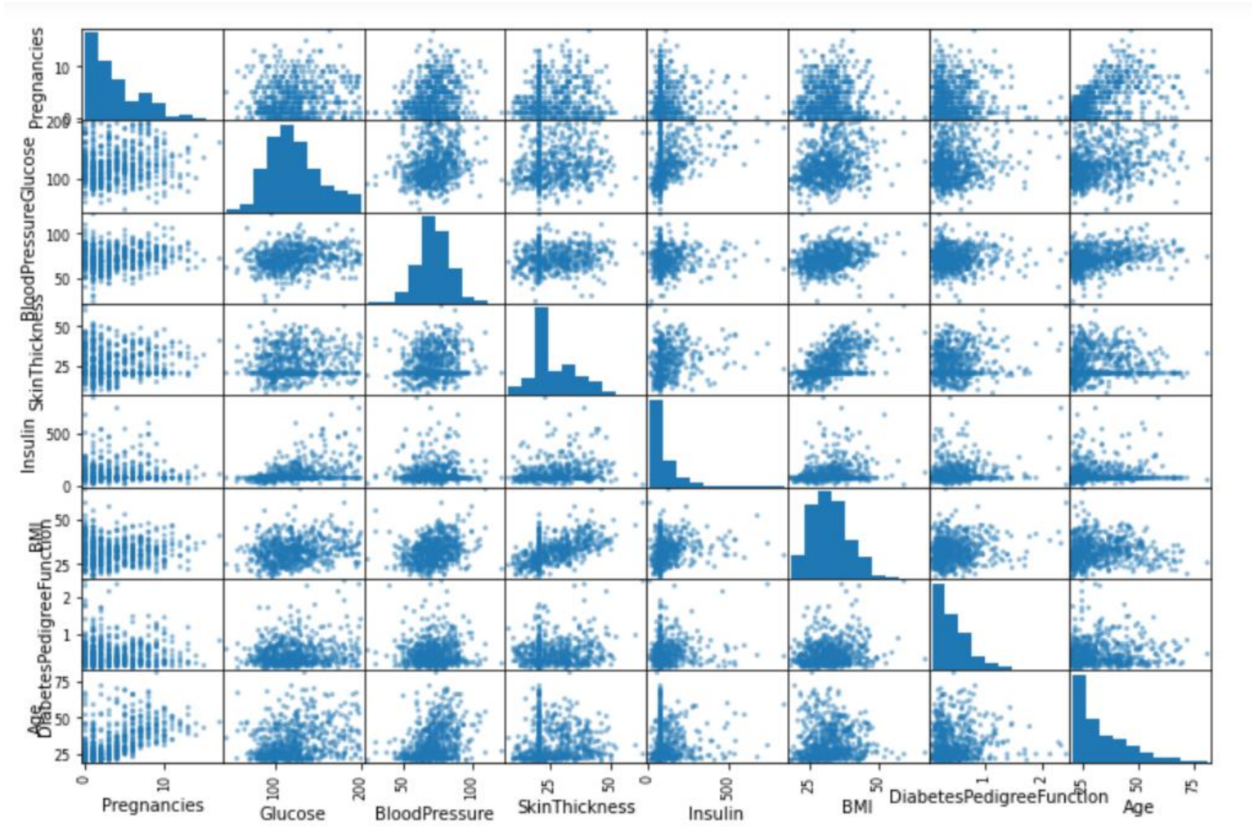
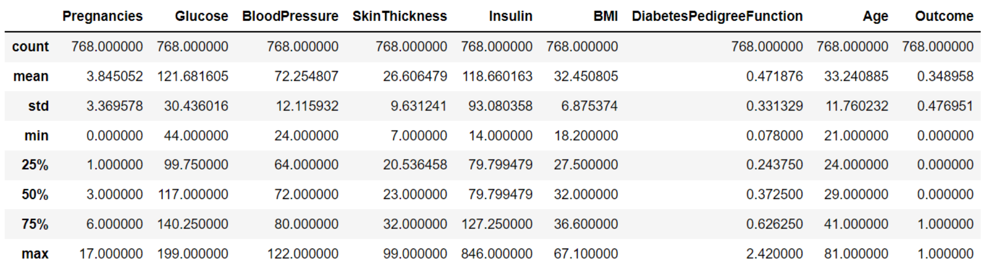


Figure 5

Figure 5 is a scatter matrix. This is one way to visualize the correlation between two attributes. Some of the values that show higher correlation are pregnancies vs age, skin thickness vs BMI, and glucose vs insulin.

Prepare the Data for Machine Learning Algorithms

The first step I took to prepare the data is to replace the infeasible values with the mean value for each column. The updated description of the data is shown in figure 6. I separated the attributes and label so that x contains the attributes and y contains the label. Next, I split the train and test data so that 20% of the data is saved for testing. I scaled the data so that all the attributes are within the same range.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.681605	72.254807	26.606479	118.660163	32.450805	0.471876	33.240885	0.348958
std	3.369578	30.436016	12.115932	9.631241	93.080358	6.875374	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	79.799479	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Figure 6

Figure 7 shows the updated distribution after replacing the zero values with the mean values. There is a large spike in the center of the blood pressure, skin thickness, and insulin graphs due to the replacement of the bad values. The data includes many people that are 21 years old so the age graph is right-skewed (positive). The pregnancies and diabetes pedigree function graphs are also right-skewed. The glucose, blood pressure, and BMI graphs look like standard bell curves.

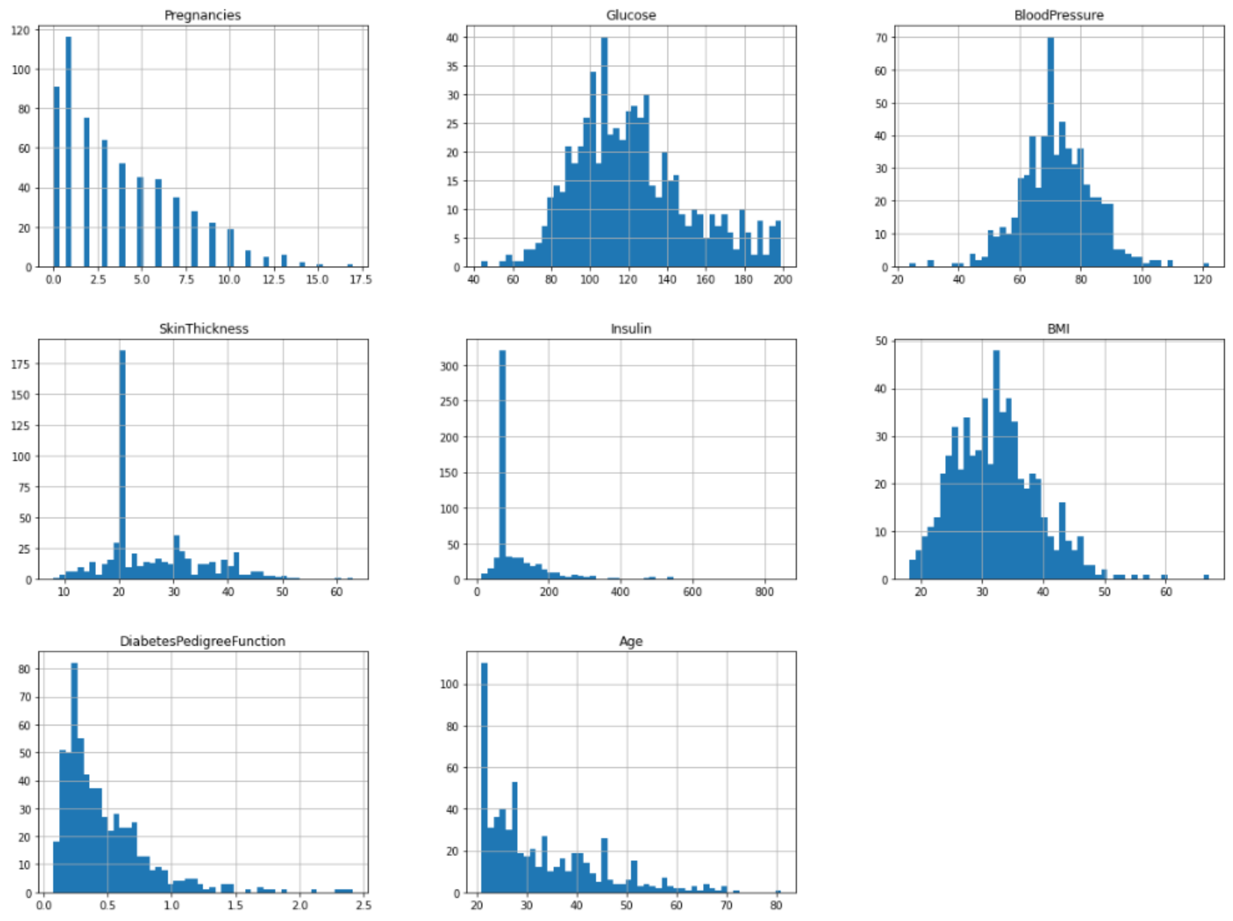
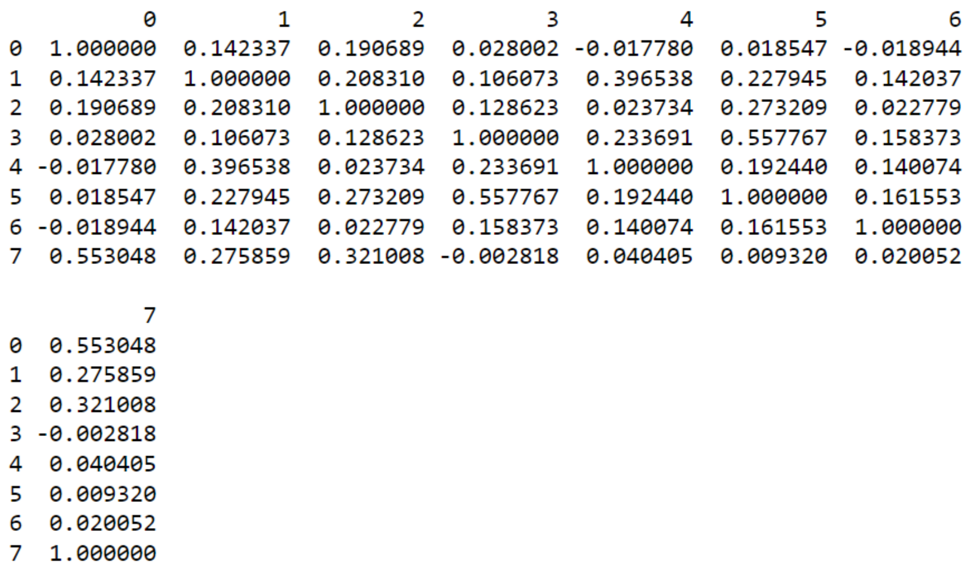


Figure 7

Using the scaled data, I created a correlation matrix shown in Figure 8. I looked at the matrix to figure out which attributes had the highest correlation with each other, and it was similar to the correlation I saw earlier in the scatter matrix. Skin thickness vs BMI had the highest correlation of 0.557767 and age vs pregnancies had the next highest correlation of 0.553. The attributes with the lowest correlation values were blood pressure and diabetes pedigree function so they were removed from the dataset. The relevant features to be used are pregnancies, glucose, skin thickness, insulin, BMI, and age.



```
          0         1         2         3         4         5         6
0  1.000000  0.142337  0.190689  0.028002 -0.017780  0.018547 -0.018944
1  0.142337  1.000000  0.208310  0.106073  0.396538  0.227945  0.142037
2  0.190689  0.208310  1.000000  0.128623  0.023734  0.273209  0.022779
3  0.028002  0.106073  0.128623  1.000000  0.233691  0.557767  0.158373
4 -0.017780  0.396538  0.023734  0.233691  1.000000  0.192440  0.140074
5  0.018547  0.227945  0.273209  0.557767  0.192440  1.000000  0.161553
6 -0.018944  0.142037  0.022779  0.158373  0.140074  0.161553  1.000000
7  0.553048  0.275859  0.321008 -0.002818  0.040405  0.009320  0.020052

          7
0  0.553048
1  0.275859
2  0.321008
3 -0.002818
4  0.040405
5  0.009320
6  0.020052
7  1.000000
```

Figure 8

At this point, the data has been split, cleaned, and is ready to be used to train a model. Some of the ways that I would improve the dataset are:

- Increase the number of instances to train the model more accurately.
- Collect more accurate data so that values do not need to be replaced.
- Increase the number of attributes such as calories per day, amount of physical activity each day, and profession. Anything that could potentially affect one’s health can help create a more accurate model.
- Include other demographics (genders, younger than 21, different ethnicities, etc.)

Explore Many Different Models and Shortlist the Best Ones

In this project, I explored the following four models:

1. Naïve Bayes
2. Decision Tree
3. KNN
4. Logistic Regression

	Naïve Bayes	Decision Tree	KNN	Logistic Regression
Accuracy Score	0.747	0.714	0.721	0.76
Precision Score	0.673	0.529	0.647	0.75
Recall Score	0.6	0.491	0.6	0.655
F1 Score	0.635	0.509	0.623	0.699
AUC Score	0.795	0.629	0.723	0.813

Figure 9

Figure 9 shows how the different models performed using several different performance metrics. Since recall score is the most important metric for this model, we will use recall to determine the strength of each model. The models ranked from best performing to worst performing are:

1. Logistic Regression
2. Naïve Bayes (Tie)
3. KNN (Tie)
4. Decision Tree

Fine Tuning with Ensemble

	Naïve Bayes	Decision Tree	KNN	Logistic Regression	Ensemble
Accuracy Score	0.747	0.714	0.721	0.76	0.747
Precision Score	0.673	0.529	0.647	0.75	0.708
Recall Score	0.6	0.491	0.6	0.655	0.618
F1 Score	0.635	0.509	0.623	0.699	0.66
AUC Score	0.795	0.629	0.723	0.813	

Figure 10

Figure 10 shows how the models performed when using multiple models as an ensemble to make binary classification decisions. I used the hard voting technique, and the recall score was about 0.618 using this method. One thing that I would have changed if I were to do this project again would be to try using the Random Forest classifier and see how it would compare to the other scores. A larger sample size would help with training accuracy and would result in a higher performance score for each model.