DISEASE X DETECTION
Maria Zalivaka

1. Project description:

Project type: The project falls under the category of disease detection or classification, where the goal is to categorize individuals into two classes: those who have "DiseaseX" (Response 1) and those who do not have the disease (Response 0)

Objective: The main objective is to develop a predictive model that can accurately classify individuals into one of two classes (Response 0 or 1) based on a set of predictor variables. The term "predict" in this context means determining the likelihood or probability of an individual belonging to a particular class.

Details of the given data set:

91 columns: 90 predictor variables, 1 response variable with values [0,1], 110,000 rows. All predictor variables are numeric.

```
Number of columns (df): 91
Number of rows (df): 109999
Number of values (df): 10009909
```

There are no Missing values in this data set.

```
Total Missing Values in df: 0
```

Percentage distribution of Response variable=0 and Response variable=1:

```
Percentage of value=1 for Response column: 45.34%
Percentage of value=0 for Response column: 54.66%
```

1. **Data pre-processing and data splitting techniques** (provide proper justification/explanation for each technique utilized in your project)

First, it's important to run an **F-statistic** test using ANOVA table:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Response   R-squared:                       0.224
Model:                            OLS   Adj. R-squared:                  0.223
Method:                 Least Squares   F-statistic:                     351.9
Date:                Fri, 23 Feb 2024   Prob (F-statistic):               0.00
Time:                        17:53:00   Log-Likelihood:                -65429.
No. Observations:              109999   AIC:                         1.310e+05
Df Residuals:                  109908   BIC:                         1.319e+05
Df Model:                          90
Covariance Type:            nonrobust
```

With F-statistic being 351.9 and Prob (F-statistic) being less than 0.00, we can say that the predictor variables indeed correlate with the response variable. Next step would be to determine which variables are insignificant - for this we have to consider the p-values for each predictor variable individually. I'm setting the p-value significance level at 0.05 Due to a large number of predictor variables we will be using **Backward Selection** to eliminate the insignificant variables (starting with all variables).

```
Variable: X7, p-value: 0.10500663666406954
Variable: X17, p-value: 0.40936504569123144
Variable: X22, p-value: 0.10231877684332615
Variable: X32, p-value: 0.25549326662193406
Variable: X35, p-value: 0.5796011919301081
Variable: X45, p-value: 0.43589585123834984
Variable: X47, p-value: 0.358672268786416
Variable: X49, p-value: 0.6156862516314043
Variable: X50, p-value: 0.23652710243142208
Variable: X52, p-value: 0.10023546034130877
Variable: X54, p-value: 0.5123495973626324
Variable: X56, p-value: 0.15707655336265267
Variable: X66, p-value: 0.1455203676152021
Variable: X67, p-value: 0.057331422308647656
Variable: X68, p-value: 0.3645172674974191
Variable: X70, p-value: 0.2066961434428566
Variable: X72, p-value: 0.9713354275368757
Variable: X73, p-value: 0.8294164250423793
Variable: X77, p-value: 0.7490107658220715
Variable: X79, p-value: 0.13419203695623602
Variable: X81, p-value: 0.9127847133922921
Variable: X86, p-value: 0.6062035003782399
```

We can see that there are 22 predictor variables that can be considered insignificant due to their p-value. Now we can create a new data frame that removes those predictor variables. Then we need to print another ANOVA table to run a new F-statistic test on the new data frame. After that, repeat the process until we don't have any insignificant variables.

```
Variable: X90, p-value: 0.05353184750048207
```

Repeat the process

```
Variable: X82, p-value: 0.05253983861831483
```

Repeat the process:

We can see that there are no variables left with the p-value of over 0.05.

Let's check the dataframe after backward selection:

```
Number of columns in df: 91
Number of columns in df4: 67
Number of columns in test_df: 66

That means that through backward selection 24 variables were taken out.
```

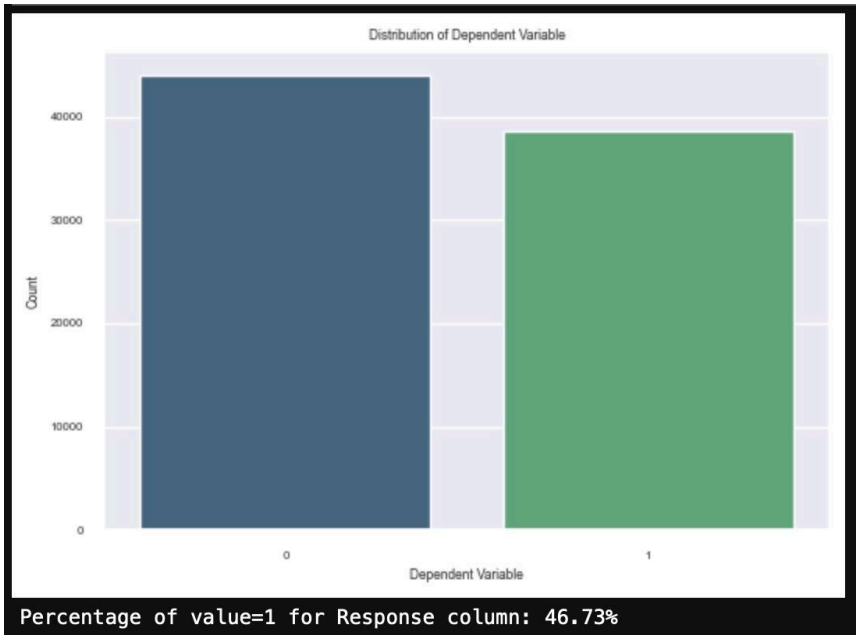
Next, let's check for **outliers** in the data. If the model relies too heavily on outliers, our prediction model will be too biased, has a risk of overfitting, and won't do well on test data. If we have outliers, we will remove them and store them in a separate data frame to test if it improved the model.

```
Number of columns: 67
Number of rows: 109999
Number of values: 7369933
Number of columns: 67
Number of rows: 82571
Number of values: 5532257

We can see that the number of rows diminished by 27428,
```

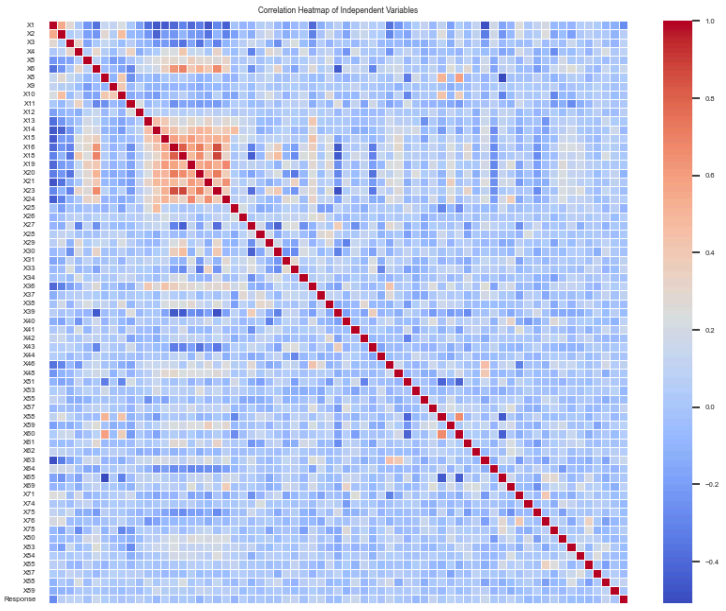
2. Data visualization

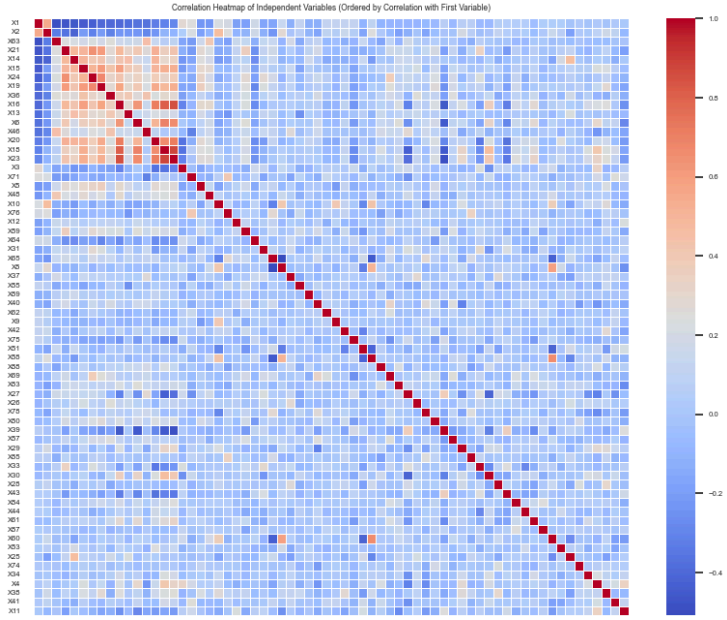
- Distribution of Dependent Variable:



We can see that 0 is slightly more prevalent in the Response (per our calculations - Percentage of value=1 for Response column: 46.73%). but since we don't know the real distribution of DiseaseX, we will have to go off those values and train our data according to those values. However, in the real world, most values would be 0 as the real % of people who have DiseaseX is a lot smaller. It changed very slightly from our earlier calculation - Percentage of value=1 for Response column: 45.34%.This is relevant because we want to be close to 45-47% for percentage of the predicted yhat value=1. If the percentage is extremely off, that might indicate issues in the classification model.

- Next we're going to take a look at correlations between independent variables and see if there is any **multicollinearity** present. First one is regular heatmap, second one is sorted based on correlation value





We can see there's multicollinearity present in the data, let's analyze this further. To do so, we will be using a **VIF score**. VIF stands for Variance Inflation Factor, and it is a measure used to quantify the extent of multicollinearity in a set of predictor variables within a regression model. Multicollinearity occurs when two or more predictor variables in a regression model are highly correlated, leading to issues in estimating the individual coefficients' impact and making the interpretation of the model less reliable. For this code we will be using a VIF threshold of 10.

```
Variables with high VIF (indicating multicollinearity):
   Variable        VIF
0        X1  46.467955
12      X14  12.709399
13      X15  15.459743
14      X16  28.116811
15      X18  43.964277
16      X19  24.390970
17      X20  29.270529
18      X21  25.284505
19      X23  41.189877
20      X24  19.872342
```

There are 10 variables with high VIF scores. In an attempt to deal with multicollinearity we will create an additional data frame that removes multicollinear variables. Then we will test whether it improved the model.

3. **Model Selection**

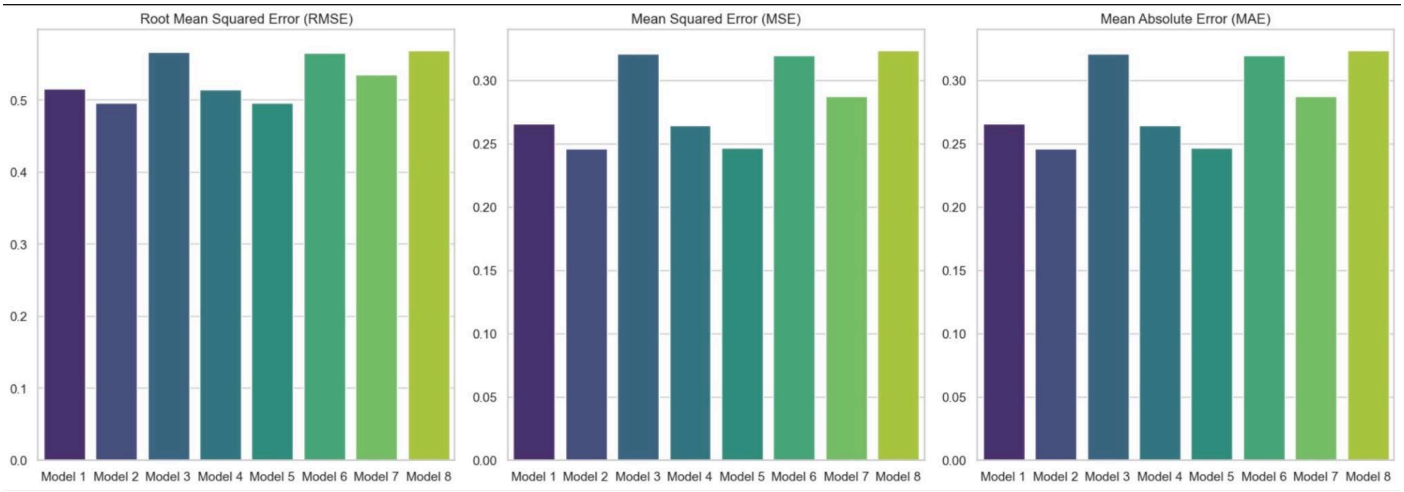
Since it's a classification project, I chose **Logistic Regression** as my model. Logistic regression models assume a linear relationship between the predictor variables and the log-odds of the outcome. Moreover, due to this algorithm detecting diseases, we want to **minimize False Negatives**. Meaning, we want to minimize cases where a diagnostic test incorrectly indicates that an individual does not have a disease when, in fact, they do. That means we want to prioritize sensitivity, meaning the test will have a lower likelihood of producing false negatives. For resampling methods I chose **Train-Test split and Stratified K-fold Cross-Validation**. In an attempt to find the best model, I ran 8 different models, they're all Logistic Regression models, however they use different resampling methods and different data splitting datasets:

Model #	Resampling Method	Data set (what data splitting technique were used)	Weights to minimize FN?	Accuracy of the model
Model 1	Train-Test Split	Backward selection	No	73.4%
Model 2	Train-Test Split	Backward selection, no outliers	No	75.4%
Model 3	Train-Test Split	Backward selection, no outliers and no multicollinear variables	No	67.9%
Model 4	Stratified K-fold, k=8	Backward selection	No	73.6%
Model 5	Stratified K-fold, k=8	Backward selection, no outliers	No	75.4%

Model 6	Stratified K-fold, k=8	Backward selection, no outliers and no multicollinear variables	No	68.1%
Model 7	Stratified K-fold, k=8	Backward selection, no outliers	Yes	71.3%
Model 8	Stratified K-fold, k=8	Backward selection	Yes	67.6%

Visualization of findings:

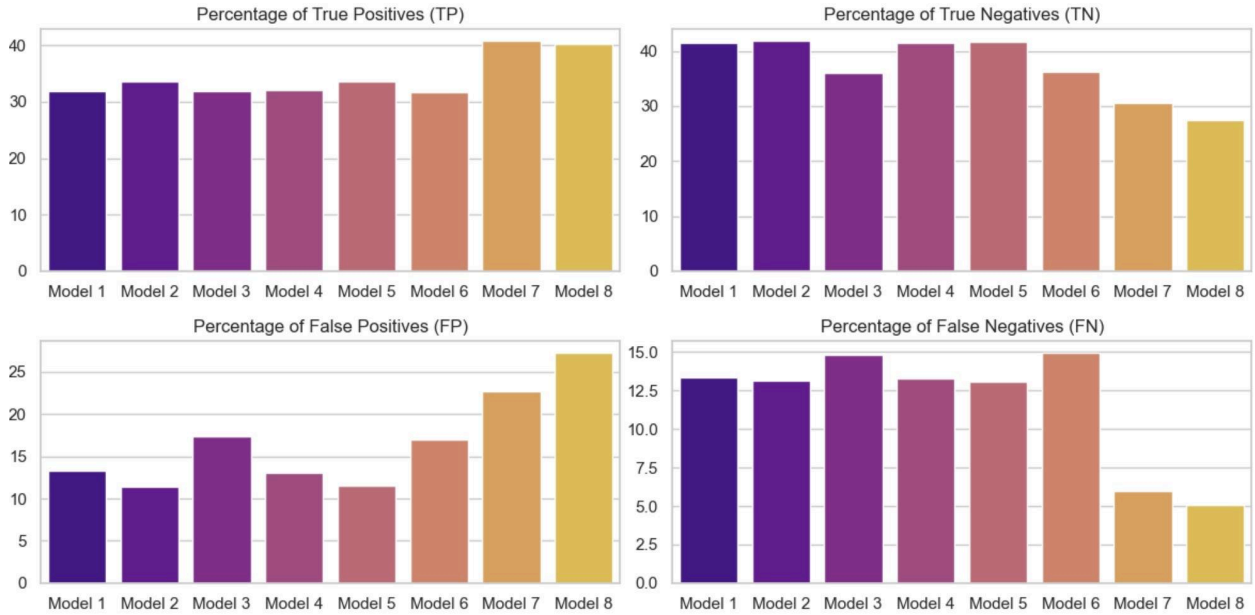
1. RMSE, MSE, and MAE for 8 models



- Models 3, 6, and 8 have the highest error values - suggesting those might be the worse choices for our final model

2. Visualizing the Confusion Matrix Values

- Since we want to minimize FN values, models 7 and 8 would be good choices based on confusion matrix values.

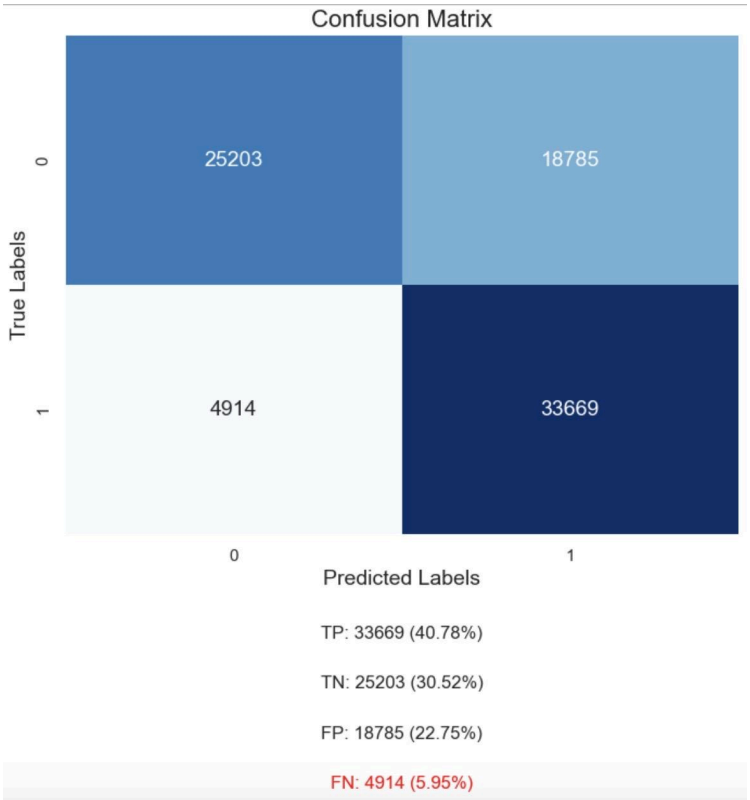


Model selected: Model 7 - Minimizing False Negatives - I chose this model to minimize FN, while maintaining a good accuracy of the whole model.

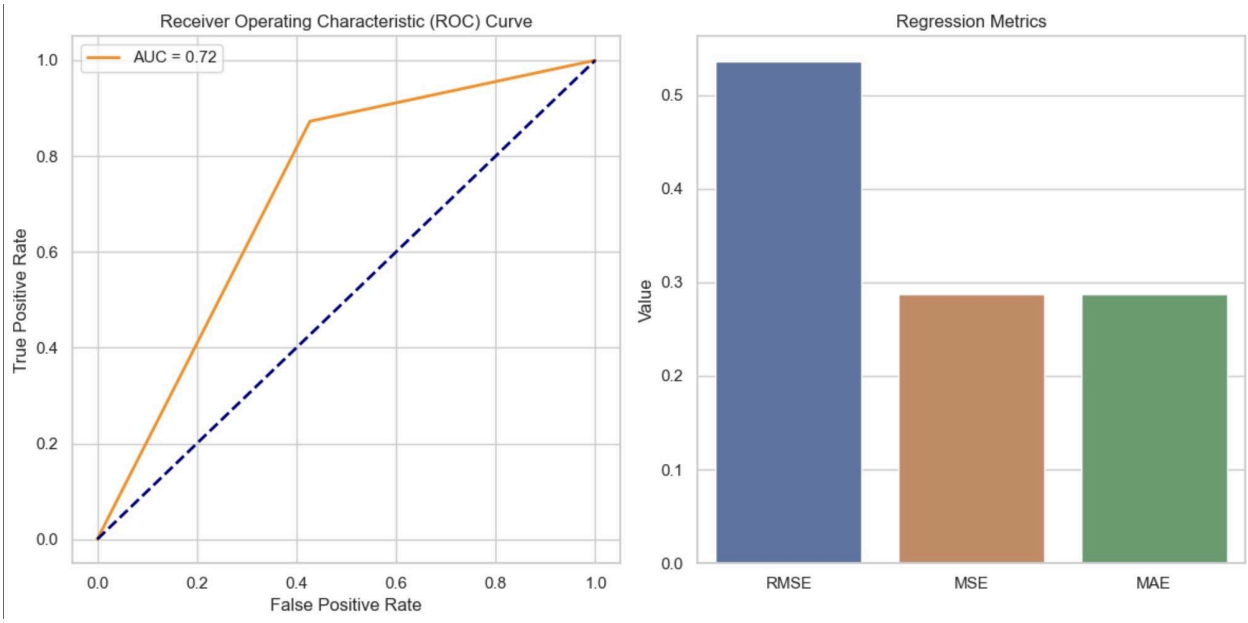
1. Model selection, resampling method, performance metric:
    a. **Model:** Logistic Regression. Reasoning: linear regression model, which can prevent overfitting and increase interpretability. I chose this model because logistic regression is commonly used for classification problems with numeric predictor variables. This model is applied on df5 - which is a dataset selected using backwards selection and also removing outliers.
    b. **Resampling method:** Stratified K-fold Cross-Validation. Reasoning: I chose this resampling method over Train-Test Split because it can reduce bias and prevent overfitting the data. Also, K-fold cross-validation provides a better estimate of how well a model generalizes to new, unseen data.
    c. **Performance metric** used to gauge the prediction accuracy of the model: model accuracy %: 71.3% - pretty good for minimizing false negatives. Confusion matrix: FN value of 5.95% - this model is good at minimizing FN. ROC curve (AUC) - the AUC is 0.72, and it also shows that it minimizes false negatives.

2. Visualizing the metrics of the model:

- Confusion Matrix:



- ROC Curve, Regression metrics (RMSE, MSE, MAE):



Conclusion and Analysis of findings: I believe this model produces such results because of a number of reasons. First, we're using a logistic regression (linear) model on a large dataset - perhaps a more complex model would do better at predicting, it could potentially capture the nonlinear relationships in the data. Also, we can see that different resampling methods (stratified k-fold vs. train-split test) didn't affect the accuracy of the model drastically, I think it's because of my choice of the linear model and most fluctuations in accuracy would come in pre-processing of the data. Interestingly, removing multicollinearity from the dataset decreased accuracy in the performance. My assumption is that those multicollinear variables are actually important predictors and removing them hurts the model performance. We can also see that the model performance slightly improved using the dataset with no outliers, achieving an accuracy of 75.39%. That means that removing outliers, indeed made the train model better, but we have to be careful as to not overfit the data.