project machine learning model for conflicting variant genomic classification

under supervision /Dr Mohamed el sayeh

Student name :sarah youssef thomas

ID:211002134

**INTRODUCTION**

ClinVar is a public database that provides information about human genetic variants. These variants are typically classified by clinical laboratories on a categorical scale that includes benign, likely benign, uncertain significance, likely pathogenic, and pathogenic. When different laboratories assign conflicting classifications to the same variant, it can create confusion for clinicians and researchers trying to determine the variant's impact on a patient's disease. our ,objective is to create a classification machine learning model to predict whether or not a variant has conflicting classifications or not

**Work pipeline**

1.first we upload the necessary libraries for our analysis

A screenshot of a computer program

Description automatically generatedA screenshot of a computer error

Description automatically generated

2. then we loaded the data set which is contain 46 column and 65187 entries from exploring data we noticed that we have columns with high percentage of nulls and columns with very low percentage of null so removing the nulls will from all data set will not be applicable

A screenshot of a computer program

Description automatically generated

3.we started checked data shape and duplications which was = zero

A screenshot of a computer

Description automatically generated

4. we creates a new DataFrame, `var\_df`, which summarizes the characteristics of each column in an existing DataFrame, `df`. The new DataFrame includes columns for the variable name, data type, missing percentage, a flag indicating whether the variable is numeric or categorical, and the count of unique values. Initially, `var\_df` is created with specified columns but no rows. The missing percentages for each column in `df` are calculated and sorted in descending order. The code then iterates through each column in `df`, determines its data type, calculates its missing percentage, and counts its unique values. Based on the data type, the variable is flagged as either numeric or categorical. These details are stored in `var\_df` using `pd.concat` to append a new row for each column. Finally, the code outputs the structure and content of `var\_df`, revealing that the DataFrame contains 46 entries with five columns: 'variable\_name', 'data\_type', 'missing\_percentage', 'flag', and 'unique\_values\_count'. The data types in `var\_df` consist of 'float64' for 'missing\_percentage' and 'object' for the other columns. The summary shows a diverse range of missing percentages and unique values for each variable, providing a comprehensive overview of the dataset's structure and characteristics.

From this we found that we have a columns that have percentage of nulls greater than 40% precent so we talked decision to remove it A screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

5.we did a heatmap to understand the correlation between the numerical values based on it and the provided information about column we took decision to remove the following columns ‘'EXON', 'CLNDISDB','Feature', 'MC','CADD\_RAW’as EXON = contains dates, not performing time series, not relevant, CLNDISB = Provides MedGen database identifiers, not relevant, Feature: Value included in consequence column, MC ‘identifier; 'CADD\_RAW:  directly related to CADD\_PHRED - only CADD\_PHRED is needed with respect to genetic mutations, it uses a scale that is easier to work with.

A white rectangular object with black text

Description automatically generated

6. then we check the nummerical and categorical values and we used still missing function to identify if we having missings

A screenshot of a computer

Description automatically generated

Then we applied still missing function to fill our missing depending on data type and we updated our data with it

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

. we also applied transformation of data type from ‘int32’ to ‘int64’

A white rectangular object with green and blue lines

Description automatically generated

Then we started handling outliers by clapping using IQR

A screenshot of a computer program

Description automatically generated

Here is our values before and after removing outliers

A graph of a graph

Description automatically generated with medium confidence

A graph of a graph

Description automatically generated with medium confidence

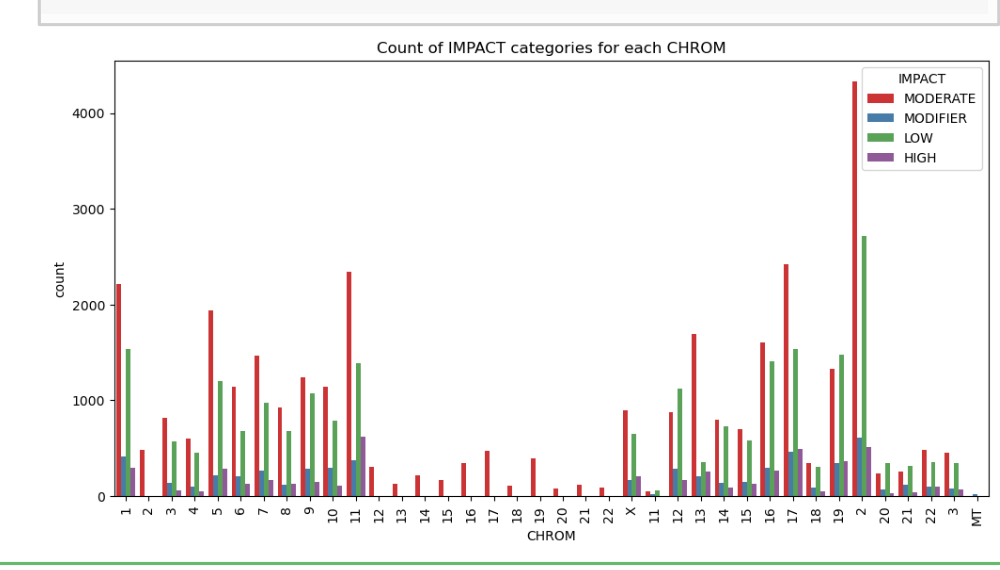
Now we have our data cleaned and with no outliers

A screenshot of a computer

Description automatically generated

**EDA Exploratory data analysis**

. 1.there are about 4,000 chromosomes that have a moderate impact identifier for   the consequence type.



2.we also did a visualization to demonstrated correlation between the 'POS', 'CADD\_PHRED columns which show datapoints frequency

A graph of a graph

Description automatically generated with medium confidence

3, this 2 visualization to confirm that we don’t have missing values

A screenshot of a graph

Description automatically generated

4, the distribution shows that the values of allele frequency are concentrated in region from 0.00 to 0.004 'AF\_ESP, 'AF\_EXAC', AF\_TGP'

A screenshot of a graph

Description automatically generated

5. a histogram that shows the distribution of all numerical columns .

A screenshot of a graph

Description automatically generated

6.number of variant in chromosomes with respect to class from that we can identify that class 0 have higher number of variants

A graph of different colored lines

Description automatically generated

7, the distribution of genes in each class we can identify that gene TNN is the most common specially in class 0

A graph with blue and orange bars

Description automatically generated

8. we also did that could to identify types of alterations between the reference and alternative alleles

A screenshot of a computer

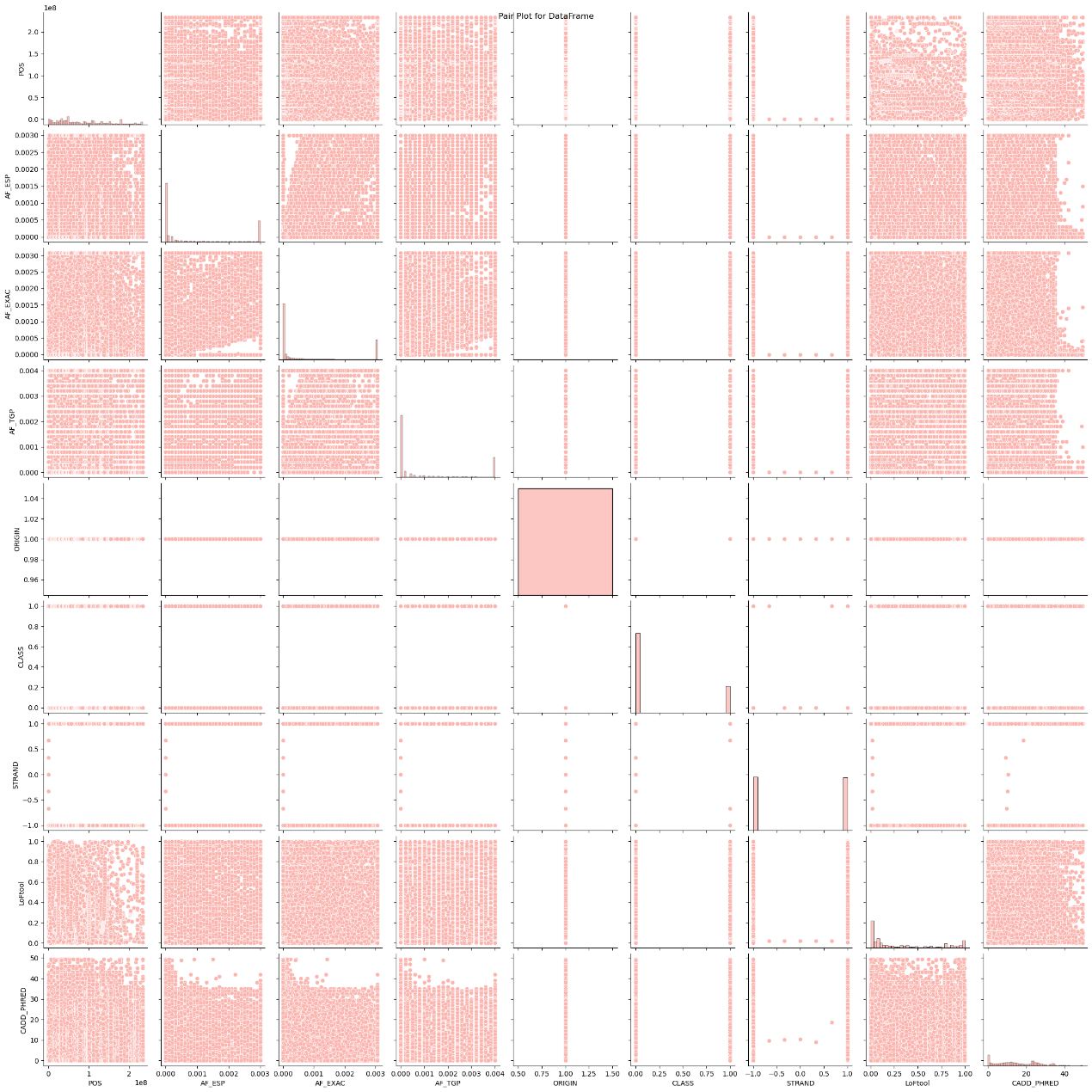
Description automatically generated

And we visualized it as following inference that indels induced alteration is more frequent in class o while single nucleotide polymorphism is more in class 1

A blue and orange rectangles

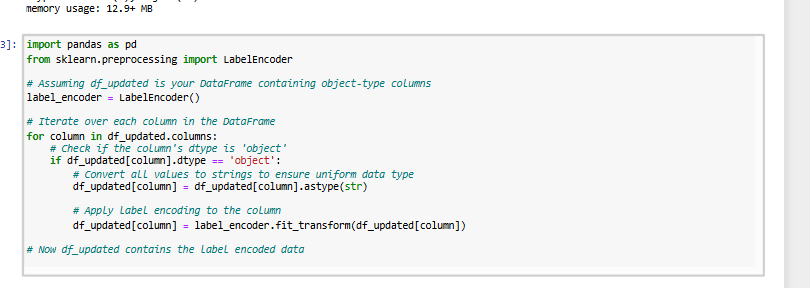
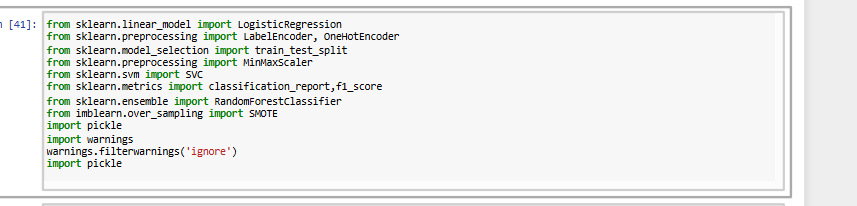
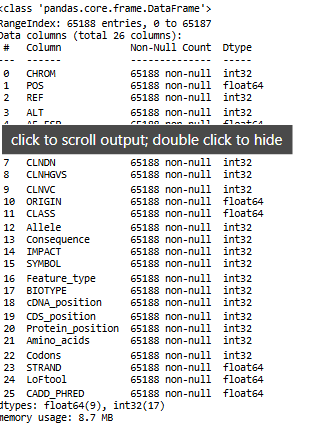
Description automatically generated

9.Finally a pairplot for the distribution of the numerical values in data



**model preprocessing and training**

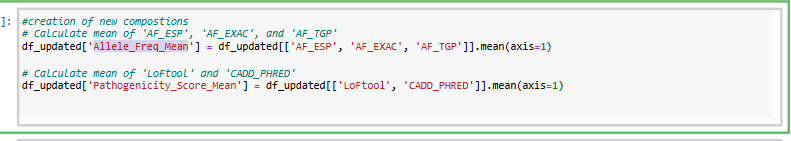
first we imported the necessary libraries and encoded the categorical values using label encoder

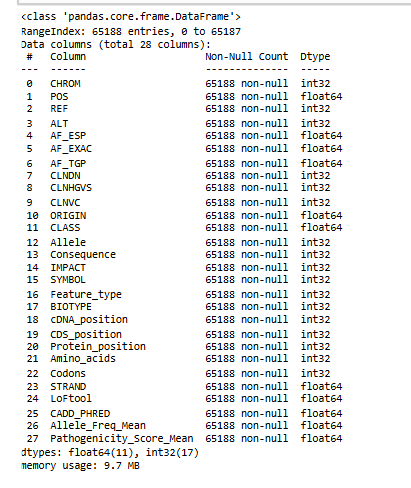


Then we created 2 new features:

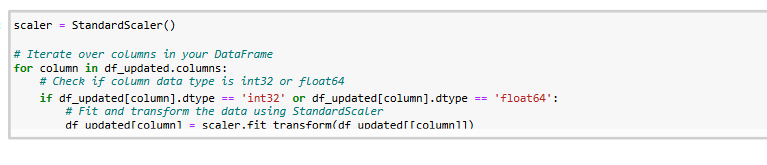
1.mean of 'LoFtool' and 'CADD\_PHRED' as 'Pathogenicity\_Score\_Mean'

2. mean of 'AF\_ESP', 'AF\_EXAC', and 'AF\_TGP' as Allele\_Freq\_Mean

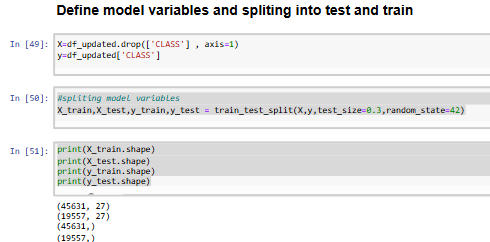




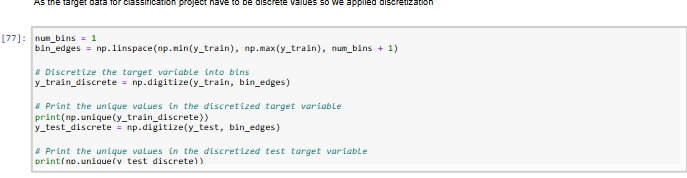
Then we applied standard scaler for all our data points as following based on iteration for data types



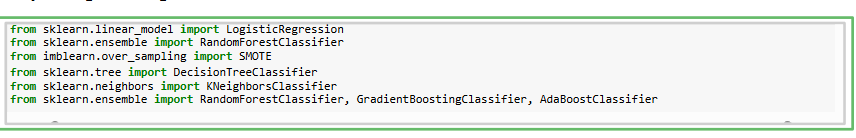
We specified the variables to be X and y and split it into train and test and checked the shape



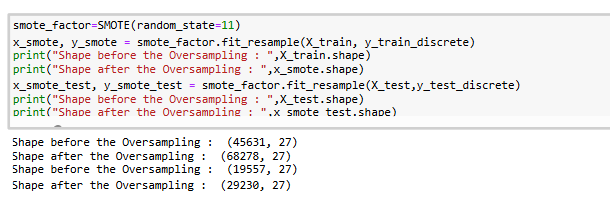
As the target data for classification project have to be discrete values so we applied discretization to ensure smooth data training



Then we imported the training models as following



We applied smote method to ensure that the categories in our data set are balanced as class o in target have higher number of values



Now we will started models training and the evolution criteria based on accuracy and classification report values and according to it

The Random Forest model shows the highest accuracy at 0.80, with a macro average f1-score also at 0.80. It exhibits strong performance in both classes, with a precision of 0.77 and 0.84 for classes 1 and 2, respectively. Additionally, it maintains a good balance between recall and precision, evidenced by high f1-scores of 0.81 for class 1 and 0.79 for class 2. This indicates that the model is proficient in both correctly identifying true positives and minimizing false positives across both classes, leading to an overall robust performance.

The Gradient Boosting model follows closely with an accuracy of 0.79 and a macro average f1-score of 0.79. It demonstrates high precision (0.80 for class 1 and 0.78 for class 2) and recall (0.78 for class 1 and 0.80 for class 2), resulting in balanced f1-scores of 0.79 for both classes. Gradient Boosting is effective in handling both minority and majority classes well, ensuring that the model is not biased towards any particular class while maintaining a high level of predictive performance.

A screenshot of a computer screen

Description automatically generated

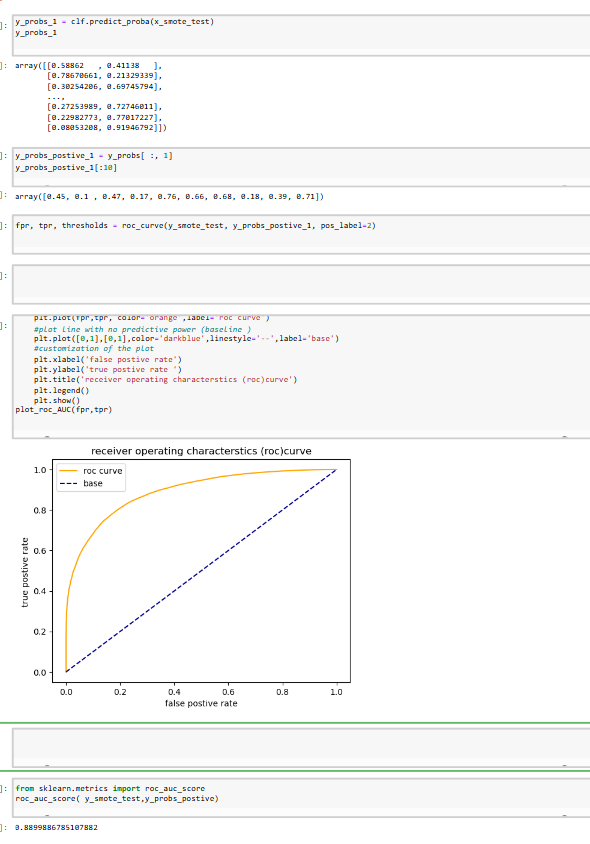
A screenshot of a graph

Description automatically generated

A screenshot of a document

Description automatically generated

We start working on GradientBoostingClassifier and random forest classifier to choose the most suitable one so we did AUC /ROC TEST from to the AUC score we found that they have the same auc\_score so we choose the random forest it have higher accuracy score ;80% and better classification report values



A screen shot of a computer

Description automatically generated

.features importance discovery

We applied features important discovery to identify which feature is the most important for our model in prediction the target variable we found that our new feature which is mean\_allele\_frequency is the heights features

A graph with colorful bars

Description automatically generated with medium confidence

Then we set a subset that contain only the heights 20 ranking features I order to increase model accuracy

A screenshot of a computer code

Description automatically generated

It give us model accuracy =80,0% but the original accuracy for model was 80,3% so we choose to complete with the original data set

Note ;working with subset will not effect the process badly as the accuracy difference is small

A white rectangular object with black lines

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Hyper parameters tunning**

We used 2 methods to search for best parameters tunning .interestingly ,we found that both give us parameter tunning that decreased the accuracy of our model so we confirmed that the best parameters in our case is the default parameters

First we set our param \_grid the for the function RandomizedSearchCV it give ide the following parameters

A screenshot of a computer

Description automatically generated

Second using gridsearchcv model

A screen shot of a computer code

Description automatically generated

So decision was taken to set our model with the default parameters states

**Model deployment on web interface**

First we downloaded our model in form of .sav file using pickle library

A green and black box

Description automatically generated with medium confidence

The in visual studio we use stream lit to create gui that takes info according to the features and display if that a case of conflicting variant classification or not

We installed stream lit created the columns inform of data frame and induced the model using the previously downloaded .sav file A computer screen shot of text

Description automatically generatedA screen shot of a computer screen

Description automatically generated

The resulted interface

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Conclusion

In our classification approach we started from data cleaning (handling missing and remove outliers ),EDA which help us to uncover patterns , scaling and encoding categorical values ,handling unbalanced data , discretization of target data and ended with random forest model with accuracy 80% we surprisingly found that the most important feature is a composite feature ;mean\_allell\_fraquency so further addition of composite features may help us to reduce data features and increase model performance.