# Ensemble Techniques

**Instructions:**

A screenshot of a cell phone

Description automatically generatedPlease share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

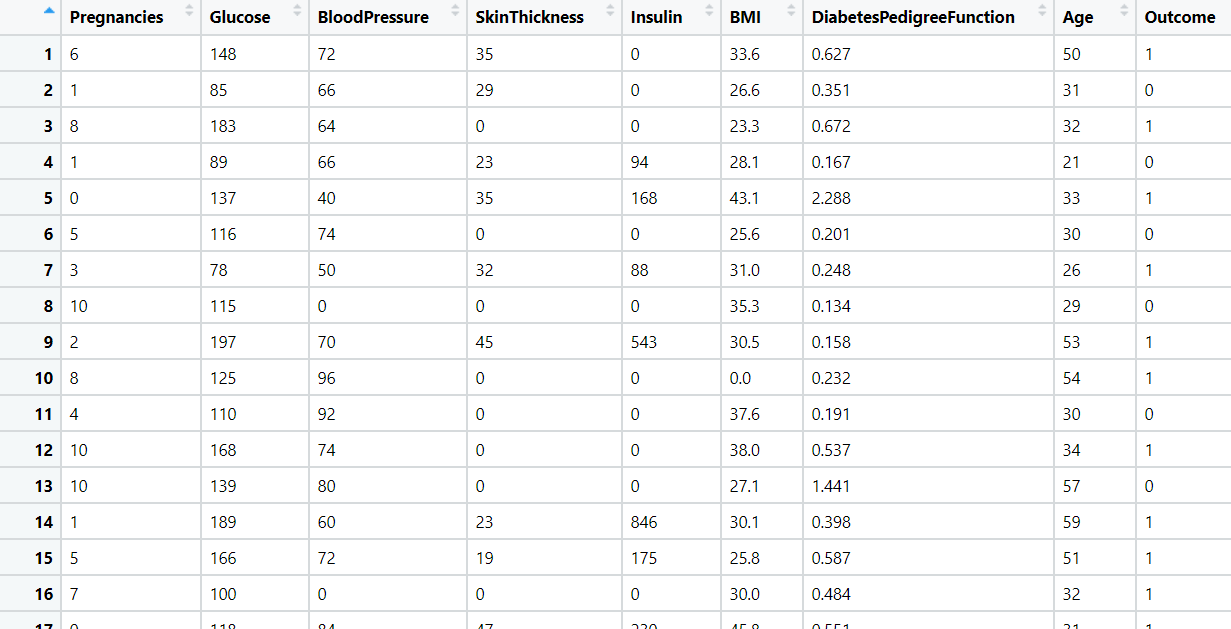
**Name: Prajay B. Urkude Batch ID:**  **16092021**

**Topic: Ensemble T****echniques**



**Problem Statements:**

1. Given is the diabetes dataset. Build an ensemble model to correctly classify the outcome variable and improve your model prediction by using GridSearchCV. You must apply Bagging, Boosting, Stacking, and Voting on the dataset. 



**Ans :- Business Objective**

To create a model which predict the new patient has diabetes or not.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| pregnancies | Number of times pregnant | Quantitative, Ratio | Relevant |
| Plasma glucose concentration | Plasma glucose concentration a 2hours in an oral glucose tolerance test | Quantitative, Ratio | Relevant |
| Diastolic blood pressure | Diastolic blood pressure (mm Hg) | Quantitative, Ratio | Relevant |
| Triceps skin fold thickness | Triceps skin fold thickness (mm) | Quantitative, Ratio | Relevant |
| 2-Hour serum insulin | 2-Hour serum insulin (mu U/ml) | Quantitative, Ratio | Relevant |
| Body mass index | Body mass index (weight in kg/(height in m)^2) | Quantitative, Ratio | Relevant |
| Diabetes pedigree function | Diabetes pedigree function | Quantitative, Ratio | Relevant |
| Age (years) | Age (years) | Quantitative, Ratio | Relevant |
| outcome | Class variable (0 or 1) | Quantitative, Nominal | Relevant |

* Import the libraries and different packages like pandas, NumPy, matplotlib, sklearn.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e., top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e., datatypes of the datasets
* Exploratory data analysis. mean, median, mode, count, min max etc.
* Dropping the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Converting the continuous data into discrete form if necessary.
* Splitting the data into train and test datasets into 80:20 ratio accordingly.

**1) GridSearchCV**

* Loading the library, from sklearn library model\_selection package import GridSearchCV function
* GridSearchCv model is hyperparameter tunning model to overcome the overfit model. Here we add the parameters Randoforestclassifier parameters and different hyperparameters like cv i.e cross validation, max\_features, min\_samples\_split to get the accuracy in the model and then initialize the GridSearchCV model and fit it into train datasets.
* We find best parameters and by taking this parameter we evaluate the model on the test datasets and again we evaluate the model on train datasets and compare the accuracy.

In our calculation we found that the test accuracy is 73% and train accuracy is 95% which is still overfit model but test accuracy is slightly greater than previous model.

**2) Bagging**

* Import the libraries, from sklearn library ensemble package import BaggingClassifier function.

We are using parameter learning rate = 0.2 and N\_estimators = 500 for adaboosting ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again evaluate the model on the training datasets and compare the accuracy.

We got 72% accuracy for the test data and 100% for the train data and the model is overfit model.

**2) Boosting**

* **Adaboost**

Import the libraries, from sklearn library ensemble package import AdaBoostingClassifier function.

from sklearn library tree package import DesicisionTreeClassifier function. We are using DecisionTreeClassifier as a base\_estimator parameter for bagging ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again evaluate the model on the training datasets and compare the accuracy.

We got 72% accuracy for the test data and 80% for the train data and the model is overfit model but the model is slightly better than previous model.

* **Gradientboosting**

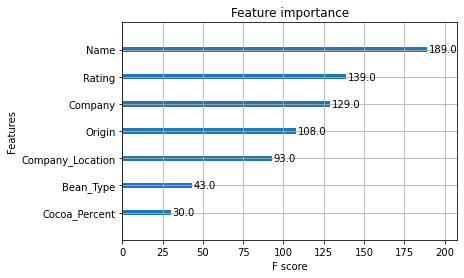
Import the libraries, from sklearn library ensemble package import GradientBoostingClassifier function.

* Fit the gradientboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 65% accuracy for the test data and 69% for the train data and the model is overfit model and have very less accuracy.

* **XGboosting**
* Import the libraries, from xgboost library import XGBClassifier function.
* Fit the XGboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 99% accuracy for the test data and 100% for the train data and the model is slightly overfit model

* We plot xgboost importance plot from which we can find the most important features for the prediction.

From the above graph we found that Name and Rating is the most important feature for the predictions.

* We apply GridSearchCV hyperparameter tuning model by using the different parameter and we found the accuracy.

Here we get 100% accuracy on test data and 99% accuracy on the train data and the model is still underfit.

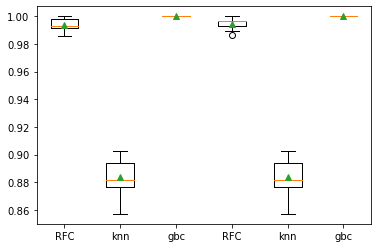
**3) Stacking :-**

* Import the libraries, and the different Classifier function to train the base learner and meta learner.

From sklearn library ensemble package import StackingClassifier. And import Kfold function from same library and model\_selection package.

K fold Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

* Creating the list of the model to evaluate. Here we are evaluating Naïve\_bayes GaussianClassifier, KNeighborsClassifier, DecisionTreeClassifier models for the base learner.
* Evaluating a given model using cross-validation
* get the models to evaluate the accuracy of the different base learner and plot the boxplot for all the three models and calculating the mean accuracy which is found out upto 76%, 75,76 respectively for all three classifier.



* make a prediction with a stacking ensemble
* Define the base learner model, we take KNeighborsClassifier, DecisionTreeClassifier, GaussianNB() classifier for trainng the base model and RandomForestClassifierClassifier for training the metalearner.
* Defining the Stacking Classifier model and fit it in the training datasets and evaluate the model in the testing dataset and testing datasets.

We got 66% accuracy for the test datasets and 82% accuracy for the training datasets and the model is overfit model.

**3) Voting :-**

* **Hard Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB , KNeighborsClassifier ,DecisionTree Classifier.
* Intializing the voting classifier by considering above three classifier .
* Fitting the voting model in the training datasets and evaluate on the test datasets.

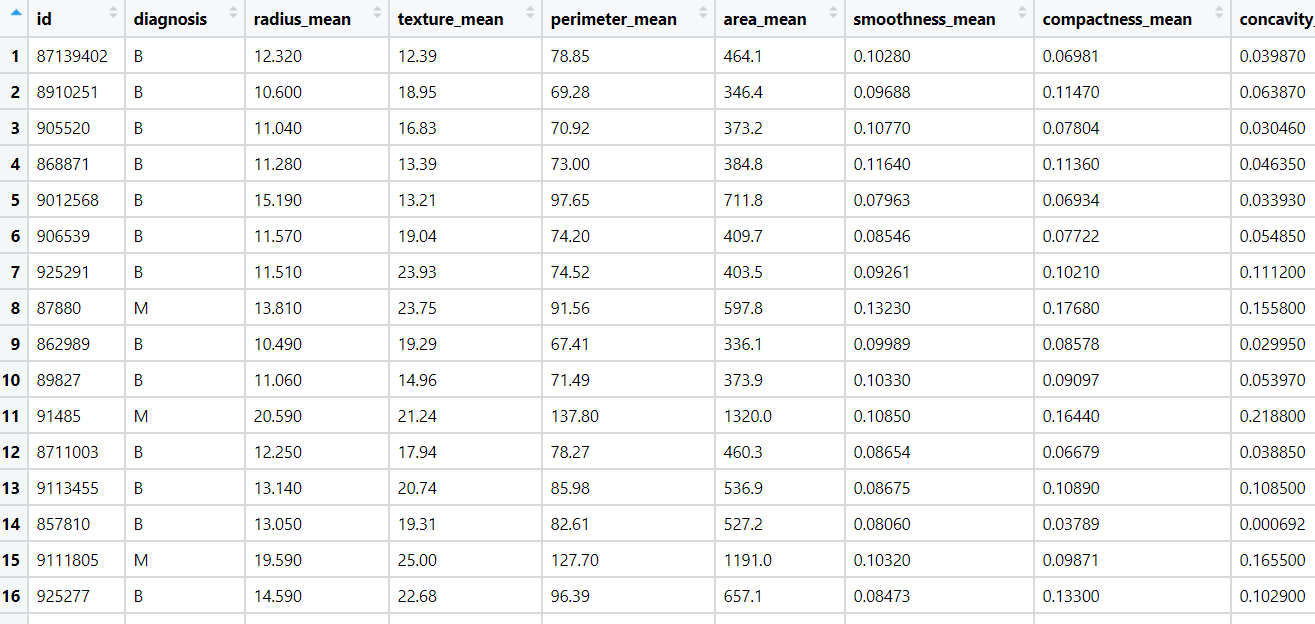
We got 71% accuracy on the test datasets and 80% accuracy on the train datasets.

* **Soft Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB() , KNeighborsClassifier() ,DecisionTree Classifier()
* Initializing the voting classifier by considering above three classifier and passing the argument as “soft”
* Fitting the voting model in the training datasets and evaluate on the test datasets.

We got 75% accuracy on the test datasets and 80% accuracy on the train datasets. Here the accuracy is slightly increase in the test dataset.

2.Most cancers form a lump called a tumour. But not all lumps are cancerous. Doctors extract a sample from the lump and examine it to find out if it’s cancer or not. Lumps that are not cancerous are called benign (be-NINE). Lumps that are cancerous are called malignant (muh-LIG-nunt). Obtaining incorrect results (false positives and false negatives) especially in a medical condition such as cancer is dangerous. So, perform Bagging, Boosting, Stacking, and Voting algorithms to increase model performance and provide your insights in the documentation.

A screenshot of a cell phone

Description automatically generated

**Ans :- Business Objective**

To create a model which predict the new patient has cancer or not

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| id | Customer id | Quantitative, Nominal | Irrelevant |
| diagnosis | Benign = no cancer  Malignant = Cancer | Qualitative, Nominal | Relevant |
| radius\_mean | Mean radius of the tumor | Quantitative, Ratio | Relevant |
| texture\_mean | Mean texture of the tumor | Quantitative, Ratio | Relevant |
| perimeter\_mean | Mean perimeter of the tumor | Quantitative, Ratio | Relevant |
| area\_mean | Mean are of the tumor | Quantitative, Ratio | Relevant |
| smoothness\_mean | Mean of smoothness of tumor | Quantitative, Ratio | Relevant |
| compactness\_mean | Mean compactness of tumor | Quantitative, Ratio | Relevant |
| concavity\_mean | Mean of concavity of tumor | Quantitative, Ratio | Relevant |
| points\_mean | Meana of points of tumor | Quantitative, Ratio | Relevant |
| symmetry\_mean | Mean of symmetry of tumor | Quantitative, Ratio | Relevant |
| dimension\_mean | Mean of dimension of tumor | Quantitative, Ratio | Relevant |
| radius\_se | Radius of the tumour | Quantitative, Ratio | Relevant |
| texture\_se | Texture of tumor | Quantitative, Ratio | Relevant |
| perimeter\_se | Perimeter of tumor | Quantitative, Ratio | Relevant |
| area\_se | Area of tumor | Quantitative, Ratio | Relevant |
| smoothness\_se | Smoothness of tumor | Quantitative, Ratio | Relevant |
| compactness\_se | Compactness of tumor | Quantitative, Ratio | Relevant |
| concavity\_se | Concavity of tumor | Quantitative, Ratio | Relevant |
| points\_se | Points of tumor | Quantitative, Ratio | Relevant |
| symmetry\_se | Symmetry of tumor | Quantitative, Ratio | Relevant |
| dimension\_se | Dimension of tumor | Quantitative, Ratio | Relevant |
| radius\_worst | Worst radius of tumor | Quantitative, Ratio | Relevant |
| texture\_worst | Worst texture of tumor | Quantitative, Ratio | Relevant |
| perimeter\_worst | Worst perimeter of tumor | Quantitative, Ratio | Relevant |
| area\_worst | Worst area of tumor | Quantitative, Ratio | Relevant |
| smoothness\_worst | Worst smoothness of tumor | Quantitative, Ratio | Relevant |
| compactness\_worst | Worst compactness of tumor | Quantitative, Ratio | Relevant |
| concavity\_worst | Worst concavity of tumor | Quantitative, Ratio | Relevant |
| points\_worst | Worst points of tumor | Quantitative, Ratio | Relevant |
| symmetry\_worst | Worst symmetry of tumor | Quantitative, Ratio | Relevant |
| dimension\_worst | Worst dimension of tumor | Quantitative, Ratio | Relevant |

* Import the libraries and different packages like pandas, NumPy, matplotlib, sklearn.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e., top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e., datatypes of the datasets
* Exploratory data analysis. mean, median, mode, count, min max etc.
* Dropping the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Converting the continuous data into discrete form if necessary.
* Splitting the data into train and test datasets into 80:20 ratio accordingly.

**1) Bagging**

* Import the libraries, from sklearn library ensemble package import BaggingClassifier function.

We are using parameter learning rate = 0.2 and N\_estimators = 500 for adaboosting ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again evaluate the model on the training datasets and compare the accuracy.

We got 97.5% accuracy for the test data and 99% for the train data and the model is overfit model.

**2) Boosting**

* **Adaboost**

Import the libraries, from sklearn library ensemble package import AdaBoostingClassifier function.

from sklearn library tree package import DesicisionTreeClassifier function. We are using DecisionTreeClassifier as a base\_estimator parameter for bagging ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again evaluate the model on the training datasets and compare the accuracy.

We got 98% accuracy for the test data and 99% for the train data and the model is overfit model but the model is slightly better than previous model.

* **Gradientboosting**

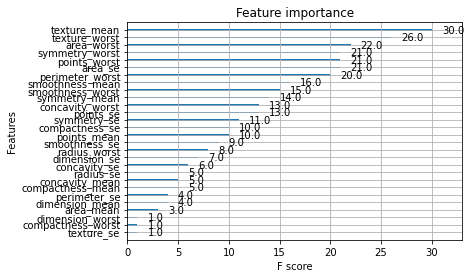
Import the libraries, from sklearn library ensemble package import GradientBoostingClassifier function.

* Fit the gradientboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 94% accuracy for the test data and 98% for the train data and the model is overfit model .

* **XGboosting**
* Import the libraries, from xgboost library import XGBClassifier function.
* Fit the XGboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 70% accuracy for the test data and 100% for the train data and the model is overfit model but the model is slightly better than previous model.

* We plot xgboost importance plot from which we can find the most important features for the prediction .

From the above graph we found that texture\_means and texture\_worst features is the most important feature for the predictions.

* We apply GridSearchCV hyperparameter tuning model by using the different parameter and we found the accuracy.

Here we get 96% accuracy on test data and 100% accuracy on the train data and the model is still overfit.

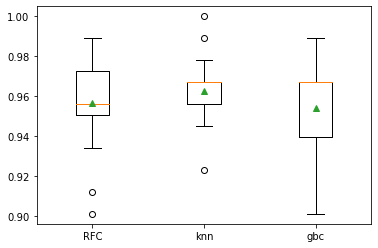
**3) Stacking :-**

* Import the libraries, and the different Classifier function to train the base learner and meta learner.

From sklearn library ensemble package import StackingClassifier. And import Kfold function from same library and model\_selection package.

K fold Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

* Creating the list of the model to evaluate. Here we are evaluating Naïve\_bayes GaussianClassifier, KNeighborsClassifier, DecisionTreeClassifier models for the base learner.
* Evaluating a given model using cross-validation
* Get the models to evaluate the accuracy of the different base learner and plot the boxplot for all the three models and calculating the mean accuracy which is found out upto 95%, 96%, 95% for the three classifier respectively.



* Make a prediction with a stacking ensemble
* Define the base learner model, we take KNeighborsClassifier, DecisionTreeClassifier, GaussianNB() classifier for trainng the base model and RandomForestClassifierClassifier for training the metalearner.
* Defining the Stacking Classifier model and fit it in the training datasets and evaluate the model in the testing dataset and testing datasets.

We got 96% accuracy for the test datasets and 96% accuracy for the training datasets and the model is perfectly fitted.

**3) Voting :-**

* **Hard Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB , KNeighborsClassifier ,DecisionTree Classifier.
* Intializing the voting classifier by considering above three classifier .
* Fitting the voting model in the training datasets and evaluate on the test datasets.

We got 97% accuracy on the test datasets and 97% accuracy on the train datasets and the model is perfectly fitted.

* **Soft Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB() , KNeighborsClassifier() ,DecisionTree Classifier()
* Initializing the voting classifier by considering above three classifier and passing the argument as “soft”
* Fitting the voting model in the training datasets and evaluate on the test datasets.

We got 96% accuracy on the test datasets and 97.5% accuracy on the train datasets. Here the model is slightly overfit.

3. A sample of global companies and their ratings are given for the cocoa bean production along with the location of the beans being used. Identify the important features in the analysis and accurately classify the companies based on their ratings and draw insights from the data. Build ensemble models such as Bagging, Boosting, Stacking, and Voting on the dataset given.

**A screenshot of a computer

Description automatically generated**

**Ans :- Business Objective**

To create a model which predict the rating of the chocolate based on the cocoa beans information.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| Company | Company of chocolate | Qualitative, Nominal | Relevant |
| Name | Name of chocolate | Qualitative, Nominal | Relevant |
| REF | Reference no. of chocolate | Quantitative, Nominal | Irrelevant |
| Review | Year from which reviews started | Quantitative, Nominal | Irrelevant |
| Cocoa\_Percent | % of cocoa in the chococlate | Quantitative, Ratio | Relevant |
| Company\_Location | Location of company | Qualitative, Nominal | Relevant |
| Rating | Rating of the chocolate | Quantitative, Ratio | Relevant |
| Bean\_Type | Type of bean | Qualitative, Nominal | Relevant |
| Origin | Origin of the bean | Qualitative, Nominal | Relevant |

* Import the libraries and different packages like pandas, NumPy, matplotlib, sklearn.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e., top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e., datatypes of the datasets
* Exploratory data analysis. mean, median, mode, count, min max etc.
* Dropping the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Converting the continuous data into discrete form if necessary.
* Splitting the data into train and test datasets into 80:20 ratio accordingly.

**1) Bagging**

* Import the libraries, from sklearn library ensemble package import BaggingClassifier function.

We are using parameter learning rate = 0.2 and N\_estimators = 500 for adaboosting ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again evaluate the model on the training datasets and compare the accuracy.

We got 99% accuracy for the test data and 99.9% for the train data and the model is good.

**2) Boosting**

* **Adaboost**

Import the libraries, from sklearn library ensemble package import AdaBoostingClassifier function.

from sklearn library tree package import DesicisionTreeClassifier function. We are using DecisionTreeClassifier as a base\_estimator parameter for bagging ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 98% accuracy for the test data and 98% for the train data and the model is good.

* **Gradientboosting**

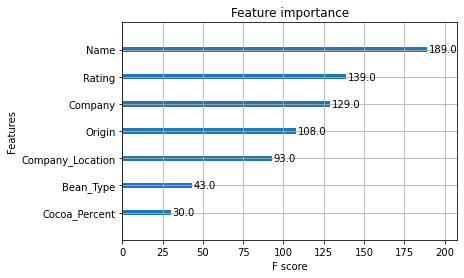
Import the libraries, from sklearn library ensemble package import GradientBoostingClassifier function.

* Fit the gradientboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 95% accuracy for the test data and 98% for the train data and the model is overfit model .

* **XGboosting**
* Import the libraries, from xgboost library import XGBClassifier function.
* Fit the XGboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 99% accuracy for the test data and 100% for the train data and the model is overfit model .

* We plot xgboost importance plot from which we can find the most important features for the prediction

From the above graph we found that Name and rating features is the most important feature for the predictions.

* We apply GridSearchCV hyperparameter tuning model by using the different parameter and we found the accuracy.

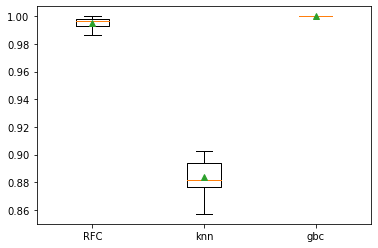
Here we get 98% accuracy on test data and 100% accuracy on the train data and the model is still overfit.

**3) Stacking :-**

* Import the libraries, and the different Classifier function to train the base learner and meta learner.

From sklearn library ensemble package import StackingClassifier. And import Kfold function from same library and model\_selection package.

K fold Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

* Creating the list of the model to evaluate. Here we are evaluating Naïve\_bayes GaussianClassifier, KNeighborsClassifier, DecisionTreeClassifier models for the base learner.
* Evaluating a given model using cross-validation
* Get the models to evaluate the accuracy of the different base learner and plot the boxplot for all the three models and calculating the mean accuracy which is found out upto 100%, 88%, 99% for the three classifier respectively.
* Make a prediction with a stacking ensemble
* Define the base learner model, we take KNeighborsClassifier, DecisionTreeClassifier, GaussianNB() classifier for trainng the base model and RandomForestClassifierClassifier for training the metalearner.
* Defining the Stacking Classifier model and fit it in the training datasets and evaluate the model in the testing dataset and testing datasets.

We got 98% accuracy for the test datasets and 100% accuracy for the training datasets and the model is overfitted

**3) Voting :-**

* **Hard Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB, KNeighborsClassifier ,DecisionTree Classifier.
* Intializing the voting classifier by considering above three classifier.
* Fitting the voting model in the training datasets and evaluate on the test datasets.

We got 97% accuracy on the test datasets and 99% accuracy on the train datasets and the model is overfitted.

* **Soft Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB() , KNeighborsClassifier() ,DecisionTree Classifier()
* Initializing the voting classifier by considering above three classifier and passing the argument as “soft”
* Fitting the voting model in the training datasets and evaluate on the test datasets.

We got 96% accuracy on the test datasets and 98% accuracy on the train datasets. Here the model is slightly overfit.

4.Data privacy is always an important factor to safeguard their customers' details. For this, password strength is an important metric to track. Build an ensemble model to classify the user’s password strength.A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**Ans :- Business Objective**

To create a model which predict the strength of the password

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| Characters | Characters of the password | Qualitative, Nominal | Relevant |
| Characters\_strength | Strength of the password  1: strong  0: weak | Quantitative, ordinal | Relevant |

* Import the libraries and different packages like pandas, NumPy, matplotlib, sklearn.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e., top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e., datatypes of the datasets
* Exploratory data analysis. mean, median, mode, count, min max etc.
* Dropping the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Divide the passwords of characters column into each column and converted into numerical dense matrix by using TfidfVectorizer function which is imported from sklearn library feature\_extraction.text package.
* Splitting the data into train and test datasets into 80:20 ratio accordingly.

**Bagging**

* Import the libraries, from sklearn library ensemble package import BaggingClassifier function.

We are using parameter learning rate = 0.2 and N\_estimators = 500 for adaboosting ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again evaluate the model on the training datasets and compare the accuracy.

We got 95% accuracy for the test data and 98% for the train data and the model is overfit.

**2) Boosting**

* **Adaboost**

Import the libraries, from sklearn library ensemble package import AdaBoostingClassifier function.

from sklearn library tree package import DesicisionTreeClassifier function. We are using DecisionTreeClassifier as a base\_estimator parameter for bagging ensemble technique.

* Fit the bagging classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 87% accuracy for the test data and 87% for the train data and the model is good but the accuracy is poor.

* **Gradientboosting**

Import the libraries, from sklearn library ensemble package import GradientBoostingClassifier function.

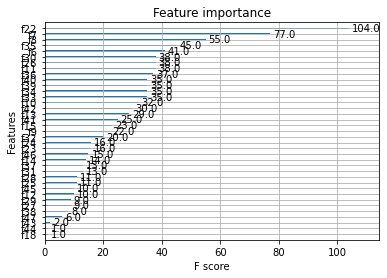
* Fit the gradientboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 86% accuracy for the test data and 86% for the train data and the model is good but the accuracy is less..

* **XGboosting**
* Import the libraries, from xgboost library import XGBClassifier function.
* Fit the XGboosting classifier model in the training dataset and evaluate the model on test dataset and find the accuracy. Again, evaluate the model on the training datasets and compare the accuracy.

We got 94% accuracy for the test data and 100% for the train data and the model is overfit model .

* We plot xgboost importance plot from which we can find the most important features for the prediction



From the above graph we found that feature2 is the most important feature for the predictions.

* We apply GridSearchCV hyperparameter tuning model by using the different parameter and we found the accuracy.

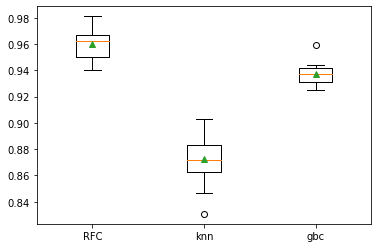
Here we get 94% accuracy on test data and 99% accuracy on the train data and the model is still overfit.

**3) Stacking :-**

* Import the libraries, and the different Classifier function to train the base learner and meta learner.

From sklearn library ensemble package import StackingClassifier. And import Kfold function from same library and model\_selection package.

K fold Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

* Creating the list of the model to evaluate. Here we are evaluating Naïve\_bayes GaussianClassifier, KNeighborsClassifier, DecisionTreeClassifier models for the base learner.
* Evaluating a given model using cross-validation
* Get the models to evaluate the accuracy of the different base learner and plot the boxplot for all the three models and calculating the mean accuracy which is found out upto 96%, 88%, 93% for the three classifier respectively.
* Make a prediction with a stacking ensemble
* Define the base learner model, we take KNeighborsClassifier, DecisionTreeClassifier, GaussianNB() classifier for trainng the base model and RandomForestClassifierClassifier for training the metalearner.
* Defining the Stacking Classifier model and fit it in the training datasets and evaluate the model in the testing dataset and testing datasets.

We got 93% accuracy for the test datasets and 99% accuracy for the training datasets and the model is overfitted

**3) Voting :-**

* **Hard Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB, KNeighborsClassifier ,DecisionTree Classifier.
* Intializing the voting classifier by considering above three classifier.
* Fitting the voting model in the training datasets and evaluate on the test datasets.

We got 93% accuracy on the test datasets and 97% accuracy on the train datasets and the model is overfitted.

* **Soft Voting**
* Import the libraries, from sklearn library ensemble package import VotingClassifier function.
* Initializing the classifier, here we are using GaussianNB() , KNeighborsClassifier() ,DecisionTree Classifier()
* Initializing the voting classifier by considering above three classifier and passing the argument as “soft”
* Fitting the voting model in the training datasets and evaluate on the test datasets.

We got 86% accuracy on the test datasets and 91% accuracy on the train datasets. Here the model is overfit.