ARE THOSE VINHO VERDE WINES TASTY?

MACHINE LEARNING METHODS IN ECONOMICS
PROF. MELANIE KRAUSE, PH.D

ESRA DAGLI
MATRICULATION NUMBER:7255733

INTRODUCTION

In this study, a Wine Quality Data Set which contains red wine and white wine samples of Vinho Verde Wines was used. Wine Quality Data Set is a relatively big data set and suitable for Classification and Regression. To make the analysis short and clean, only white wine samples were utilized for this term paper. The main purpose of the machine learning analysis was to predict wine quality with given sample features. Due to privacy issues, only physicochemical and sensory variables are available for explaining the quality of the wines. The reason behind using machine learning techniques is that we can automatize this quality prediction, whenever we have new wines(observations) we can predict their quality much faster and in a convenient way. To predict qualities, 3 Classification methods were applied, Decision Tree, Random Forest, and Support Vector Machine. All the methods had more than 70% test score.

This paper contains information about descriptive statistics, the application of three methods, and the evaluation of the machine learning methods.

1)DESCRIPTIVE STATISTICS

To be able to understand the characteristics of the data and to perform the further application first, the data was imported and turned to data frame manually, later libraries were imported.

```
>>import numpy  as np

>>Import matplıtlib.pyplot as plt

>>Import sklearn as sk

>>Import pandas as pd
```

To show the size and the features following two codes were performed.

```
>>print('Shape of the original file', white_wine.shape)

Shape of the original file (4898, 12)

>>print('Columns of the original file', white_wine.columns)
```

Columns of the original file Index: ['fixed acidity', 'volatile aciditiy', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sufur dioxide', 'density', 'pH', 'sulphates', 'quality'], dtype=object)

White wine samples have 4898 observations (rows) and 12 features (columns). Features can be seen under the "print(*'Columns of the original file'*, white_wine.columns)" code. Most of them are physicochemical ingredients, due to privacy issues this data set do not contain any information about grape types or other indicators of quality.

To describe the data set a bit better, the described command was used and with this code we able to see the values of some features and their statistical characteristics.

```
>>white_wine.describe
```

	Fixed acidity	Volatile acidity		sulphates	alcohol
Count	4898.000000	4898.00000	…..	4898.00000	4898.00000
mean	6.854788	0.278241	…..	0.489847	10.514267
std	0.843868	0.100795	…..	0.1141126	1.230621
min	3.800000	0.080000	…..	0.220000	8.00000
25%	6.300000	0.210000	…..	0.410000	9.500000
75%	7.300000	0.310000	…..	0.55000	11.40000
Max	14.20000	1.100000	…..	1.080000	14.20000

When we look at the table in more detail, we can see some variables for fixed acidity, volatile acidity, sulphates, and alcohol. We can easily say that there are no missing values for these features because we see that they have 4898 observations. Before making a general assumption, we should also check the missing values inside the whole data.

To check whether we have a missing value or not, the following code was run.

```
>>white_wine.isnull().sum()
```

Fixed acidity	0
Volatile acidity	0
Citric Acid	0
Residual sugar	0
Chlorides	0
Free sulfur dioxide	0
Total sulfur dioxide	0
Density	0
pH	0
Sulphates	0
Alcohol	0
Quality	0
Dtype:int64	

This table tells us there are no missing values in this data set. we can continue our analysis without running any code for missing values. We do not have anything to drop in our dataset.

1.1) HISTOGRAMS

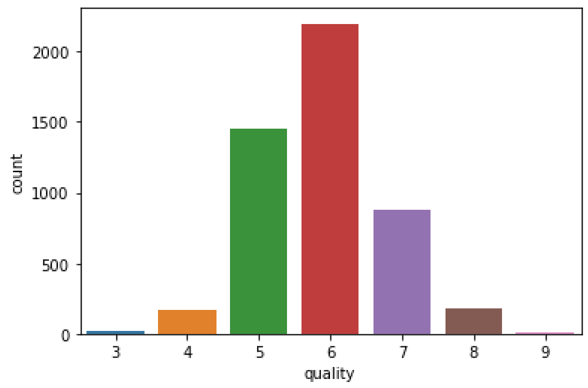
Histograms help us to visualize the data that we are working on. For this purpose, histograms for the outcome variable quality and the features were plotted.

- *OUTCOME VARIABLE- QUALITY*

To illustrate the distribution of the outcome variable 'quality' following codes were run.

plt.figure(figsize=5,5)):

ax=sns.countplot(x=*'quality'*, data=white_wine)

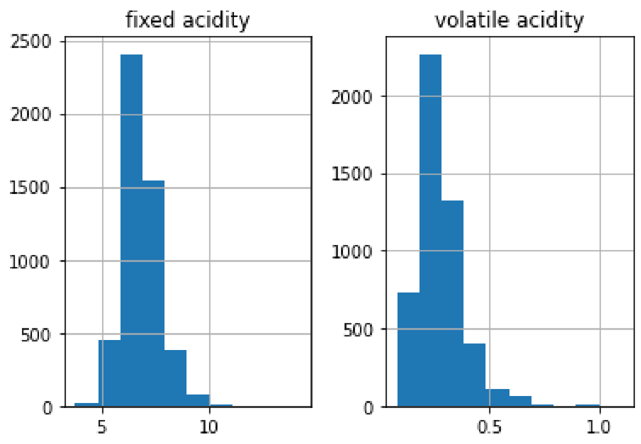


One thing that can be seen from the figure that most of the white wine samples have the quality of 5, 6, 7. They are almost more than 80% of all white wine samples.

- *FEATURES*

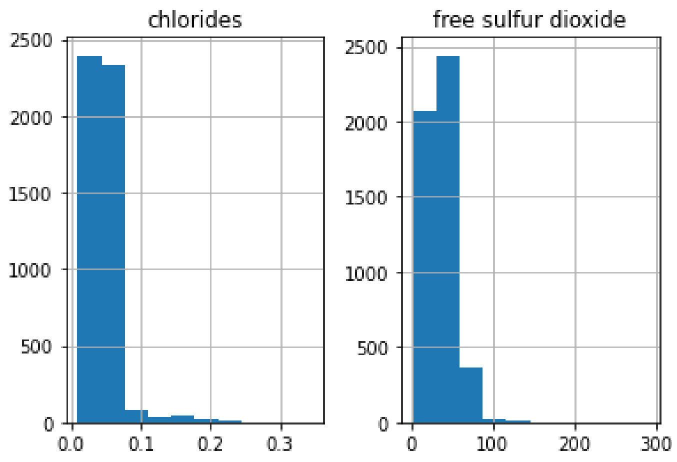
To plot the histograms for the features the following codes were performed.

>>Hist=X.hist([*'fixed acidity'*, *'volatile acidity'*])

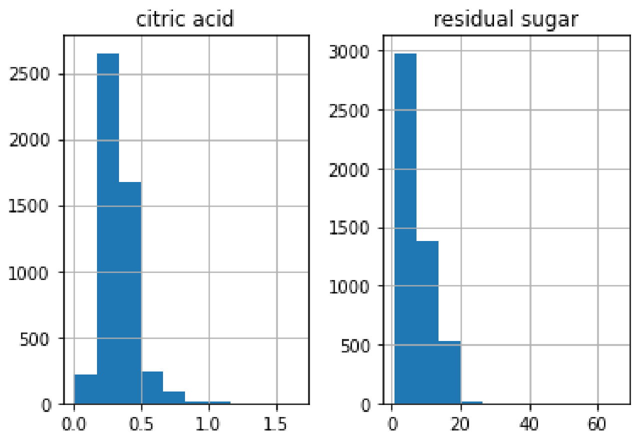


Most of the observations get a fixed acidity value between 5 and 10. Same thing almost true for volatile acidity as well. Most of the wines have a degree of volatile acidity between 0 and 0.5. For both features, there is no major variation.

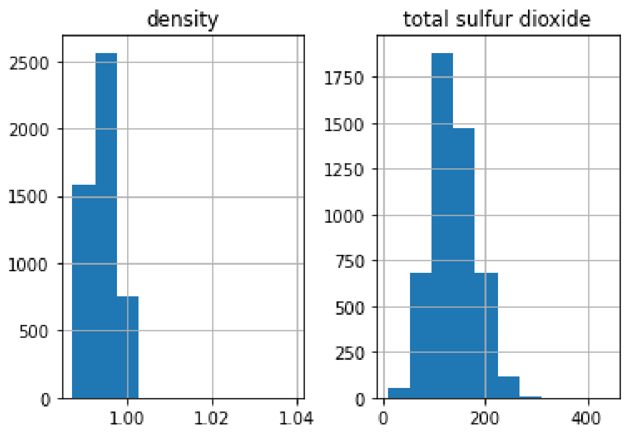
```
>>Hist=X.hist([ ‘chlorides’, ‘free sulfur dioxide’]])
```



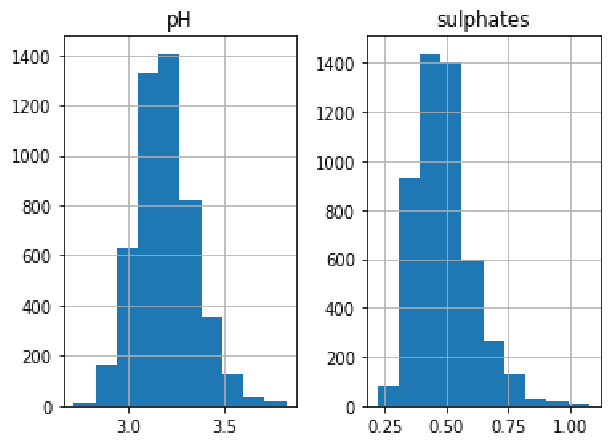
```
>>Hist=X.hist([‘citric acid’, ‘residual sugar’])
```



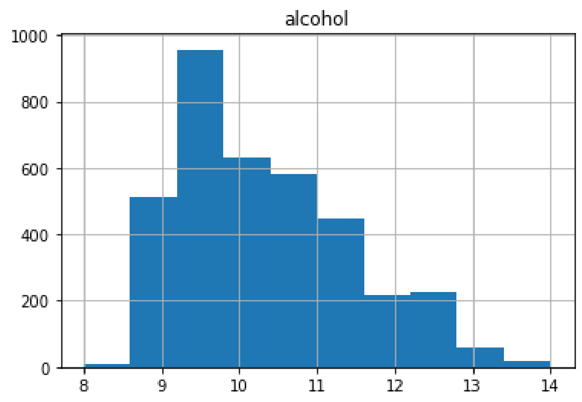
```
>>Hist=X.hist([‘total sulfur dioxide’, ‘density’])
```



```
>>Hist=X.hist([‘pH’, ‘sulphates’])
```



```
>>Hist=X.hist([‘alcohol’])
```



The histograms showed that for features there are no significant variation. Most of the white wine samples have the same degree of physicochemical actives as an ingredient. Only alcohol, pH, and sulphates look variates between observations.

1.1) DATA SPLIT

To start performing the models, first, we must create the x and y variable. To make this term paper a bit easy and well designed, Categorical variables were created for the outcome variable quality which was graded between 0 and 10 in the original data. Instead of using all the categories (0-10), 2 categories ‘good’ and ‘bad’ were generated for white wine qualities. The wines have quality more than 6 were categorized as good, else were classified as bad.

```
>> white_wine['quality']=['Good' if x>6 else 'Bad' for x in white_wine['quality']]

>>y=white_wine[ ‘quality’]

>>X=white_Wine.drop(‘quality’, axis=1)
```

We need additional data to be sure that the accuracy score we get after utilizing the machine learning methods, will be the same whenever we use the new data. If we just use a simple train and test set split, we exploit the test set for tunning the data, then we cannot use the test set for interpreting model performance with new data. Because we would like to avoid this problem, a three-wise data split was utilized throughout performing the models.

```
>>from sklearn.model_selection import train_test _split

>>X_trainval, X_test, y_trainval, y_test= train_test_split(X, y, random_state=42)

>>X_train, X_valid, y_train, y_valid=train_test_Split(X_trainval, y_trainval, random_state=42)

>>Print(X_train.shape)

(2754,11)

>>Print(X_valid.shape)

(919, 11)

>>Print(X_test.shape)

(1225,11)
```

2)METHODS

After checking the histograms, we can assume that not all the variables are relevant for predicting wine quality. As a result of that, we need to do the feature selection. In this term paper, tree-based methods, Decision Tree Classifier, and Random Forest Classifier were used to predict the categories of the white wines because they have a self- future selection. Moreover, in the data we have more good wines than bad wine. Hence, we suffer from imbalanced data which means that we need to utilize more powerful and robust methods like Support Vector Machines as an additional method to find the best accuracy score. Since we do not certainly know our futures have a linear or nonlinear effect on the outcome variable. So, the Support Vector Machine was run with 2 different kernel method, linear and Gaussian Kernel (rbf).

Throughout operating the model, grid search and cross-validation were used together because in every single model we wanted to find the best parameter or parameters and we also wanted to use cross-validation. Combining them was the practical way to perform the models. In all 4 models, to fit of the best parameter combination 10 folds were used. In the end, we averaged out the k accuracy scores.

For Decision Tree and Random Forest, Feature scaling was not used. We know that Decision trees and ensemble methods do not require feature scaling to be performed. Also, in this data set scaling reduced the accuracy scores of Random Forest and Decision Tree. Since we continued our analysis without feature scaling in Random Forest Classifier and Decision Tree Classifier.

However, for linear and Gaussian kernel Support Vector Machines using feature scaling is important, for this reason, Robust Scaler was used. Besides, Robust Scaler gave us high accuracy score rather than Standard Scaler. It showed that even though, we recategorized the outcome variable, we might still suffer from an outlier problem.

2.1) DECISION TREE CLASSIFIER

To perform a decision tree classifier with grid search and cross-validation following codes were run.

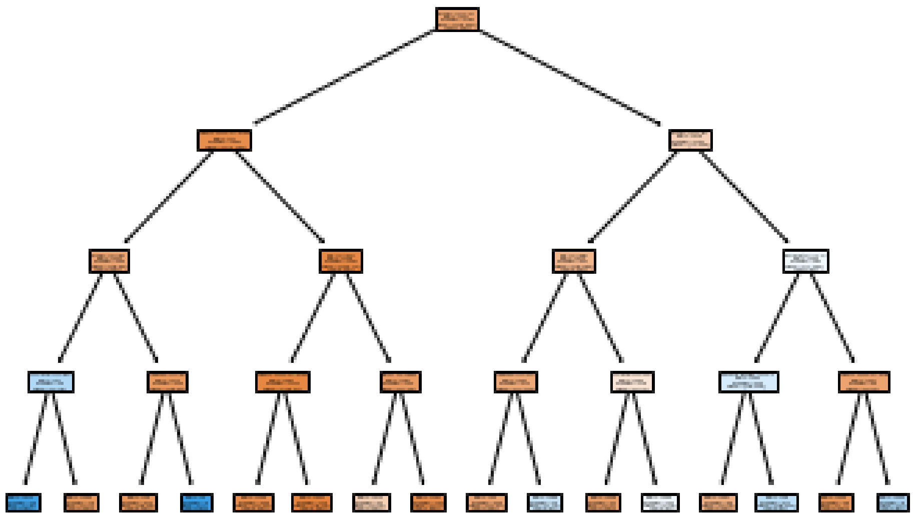
In grid search, we tried the find the max depth that gives us the best accuracy score.

```
>>From sklearn.model_selection import cross_val_score
>>From sklearn.tree import DecisionTreeClassifier
>>Best_score=0
>>For max_depth_tree in [1,2,3,4,5,6,7,8,9,10]:
    Tree=DecisionTreeClassifier(max_depth=max_depth_tree)
    Scores_tree=cross_val_score(tree, X_trainval, y_trainval, cv=10)
    Score_tree=np.mean(scores_tree)
    If score_tree> best_score:
        Best_score= score_tree
        Best_parameter_tree={ ‘max_depth’: max_depth_tree}

>>Print(‘Best parameter:’, best_parameter_tree)
Best parameter: {‘max_depth’: 4}
>>Tree=DecisionTreeClassifier(**best_parameter_tree)
>>Tree.fit(X_train, y_train)
DecisionTreeClassifier(max_depth=4)
>>Trest_score_tree=tree.score(X_test, y_test)
>>Print(‘Test set score with best parameter:’, test_score_tree)
Test set score with best parameter: 0.7975510204081633
```

In the end, for max depth is 4 and 10 folded cross-validation, we got almost 80% test score.

We can also show the visualization of the tree.



When we look at the visualization of our tree, we can see that our tree was split 4 times.

2.2) RANDOM FOREST

The second method that we used was Random Forest Classifier. We tried to find max depth and the number of estimators which gave us better performance with the grid search. Again 10 folds were used for cross-validation. To perform the random forest classification the following codes were used.

```
>>From sklearn.ensemble import RandomForestClassifier

>>For Max_depth_forest in [1,2,3,4,5,6,7,8,9,10]:

      For n_estimators in [1,10,100]:

      Forest=RandomForestClassifier(max_depth=max_depth_forest,
n_estimators=n_estimators_forest)

     Scores_forest=cross_val_score(forest, X_trainval, y_trainval, cv=10)

     Score_forest=np.mean(scores_forest)

     If score_forest > best_score:

        Best_score= score_forest

        Best_parameters_forest={'max_depth':max_depth_forest,

'n_estimators':n_estimators_forest}

>>Print('Best parameters:', best_parameters_forest)
```

```
Best parameters: {'max_depth': 10, 'n_estimators': 100}

>>Forest=RandomForestClassifier(**best_parameters_forest)

>>Forest.fit(x_trainval, y_trainval)

RandomForestClassifier(max_depth=10, n_estimators=100)

>>Test_score_forest=forest.score(X_test, y_test)

>>Print('Test set score with best parameters:', test_score_forest)

Test set score with best parameters: 0.8612244897959184
```

With random forest classifier, we got 86% as a test score which was the best score we got so far.

We can also show which that which features are important for the performance after running the following codes,

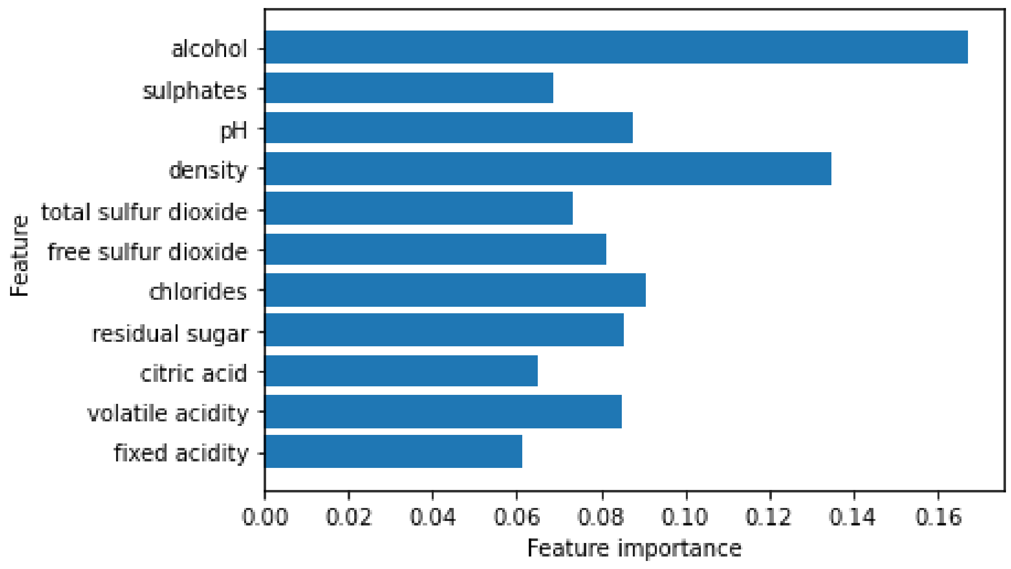
```
n_features = len(feature_names)

plt.barh(range(n_features), forest.feature_importances_)

plt.yticks(np.arange(n_features), feature_names)

plt.xlabel("Feature importance")

plt.ylabel("Feature")
```



When we look at the Feature Importance figure, we can see that the most important features are alcohol and density.

Using Random Forest and Decision Tree for predicting wine quality gave us the ability to compare prediction power of a single tree and many trees. In a decision tree, the performance depends on one individual tree, but in Random Forest includes many trees each based on a random sample of training data. Theoretically, we can say that Random forest more accurate than a Decision tree, and our analysis supports this theoretical fact. In decision Tree method we got 0.7975510204081633

While the test score in Random Forest was 0.8612244897959184. To conclude, we can say that Random Forest performs better than Decision Tree for our White wine samples data.

2.3)SUPPORT VECTOR MACHINES

To overcome the problems of the imbalanced data, a Support Vector Machine was used in this paper. Because we did not know that our white wine samples had a linear influence on the outcome variable or not, we run the support vector machine method we a different technique. The kernels were linear and radial basis function for this method.

Before running support vector machine codes, first, we did the feature scaling with robust scaler. Standard scaler also was used in the first steps of analysis as an alternative. However, Robust Scaler increased the performance of both Support Vector Machine techniques.

```
>>From sklearn.preprocessing import RobustScaler

>>Scaler=RobustScaler()

>>X_trainval_scaled=scaler.fit_trasform(X_trainval)

>>X_test_scaled=scaler.fit.transform(X_test)

>>X_train_scaled=scaler.fit_transform(X_train)

>>X_valid_Scaled=scaler.fit_transform(X_valid)
```

2.3.1) LINEAR SUPPORT VECTOR MACHINE

Now we assume that we have a linear relationship between our features and the outcome variable quality. To predict the wine quality, we run the support vector machine codes with kernel= linear option.

We know that the support vector machine is sensitive to the choice of the parameter, for this reason, again we used grid search and cross-validation together. We run the following codes to perform Support Vector Machine (linear):

```
>>From sklearn.svm import SVC

>>Best_score=0

>>For gamma_linear in [0.01,0.1,0.5,1,5,10]:

    For C_linear in [0.01, 0.1, 1, 10]:

    Svm_linear= SVC(gamma=gamma_linear, C=C_linear, kernel='linear']

    Scores_linear=cross_Val_score(svm_linear, X_trainval_scaled, y_trainval, cv=10)

    Scores_linear=np.mean(scores_linear)
    If score_linear> best_score:
      Best_score= score_linear
      Best_parameters_linear={'C': C_linear, 'gamma': gamma_linear}
>>Print( 'Best parameters:', best_parameters_linear)
Best parameters: {'C': 0.01 , 'gamma':0.01 }
>>Svm_linear=SVC(**best_parameters_linear)
>>Svm_linear.fit(X_trainval_scaled, y_trainval)
>>Test_score_linear=svm_linear.score(X_test_scaled, y_test)
>>Print('Test set score with best parameters:', test_score_linear)
Test set score with best parameters: 0.7681632653061224
```

From Linear Support Vector Machine method, we got 77% test score. This score is worse than the score of Decision Tree and Random Forest. But still, we should show the test score under radial basis function technique.

2.3.2) SUPPORT VECTOR MACHINE/ GAUSSIAN KERNEL

Now we assume that we have a nonlinear relationship in our data. For this reason, we used the Kernel=rbf option and the following codes were used.

```
>>Best_score=0
>>For gamma_rbf in [0.01, 0.1, 0.5,1,5,10]:
    For C_rbf in [0.01,0.1, 1, 10]:
    Svm_rbf=SVC(gamma=gamma_rbf, C=C_rbf, kernel='rbf')
    Scores_rbf=cross_val_score(svm_rbf, X_trainval_scaled, y_trainval, cv=10)
    Score_rbf=np.mean(scores_rbf)
    If score_rbf>best_Score:
       Best_score=score_rbf
       Best_parameters_rbf={'C'=C_rbf, 'gamma'=gamma_rbf)


>>Print('Best parameters:', best_parameters_rbf)
Best parameters: {'C':10, 'gamma':1}
>>Svm_rbf=SVC(**best_paramete_trainval)
SVC(C=10, gamma=1)
>>Test_score_rbf=svm_rbf.score(X_test_Scaled, y_test)
>>Print('test set score with best parameters:', best_parameters_rbf)
```

Test set score with best parameters: 0.8546938775510204

With the Gaussian kernel, we got better test results than the previous linear SVM. This shows that we have a nonlinear relationship in our data. Using the radial basis function is a better technique for predicting the wine quality rather than the linear version.

To sum up, we can say that the model which gives us the best performance is Random Forest Classifier. However, Support Vector Machine with Gaussian Kernel has almost the same test score for ending up with a conclusion, we should check the results of Confusion matrices and Sensitivity/ Specificity values.

3)EVALUATION

3.1)CONFUSION MATRIX

Now we can compare our models according to Confusion Matrices, sensitivity, and specificity. The following codes were used to create these evaluation criteria.

```
>>Form sklearn.metrics import confusion_matrix

>>Models=[tree,forest]

    For i in models:

    Print(i)

    Prediction=i.predict(X_test)

    Print(confusion_matrix(y_test, prediction))

    Mat1=confusion_matrix(y_test, prediction)

    Sens=mat1[1,1]/mat1[0,1] +mat1[1,1])

    Print(‘sensitivity:’, sens)

    Spec=mat1[0,0]/ (mat1[0,1]+mat1[0,0])

    Print(‘specificity:’, spec)

>>Models=[svm_linear, svm_rbf]

    For i in models:

    Print(i)

    Prediction=i.predict(X_test)

    Print(confusion_matrix(y_test, prediction))

    Mat1=confusion_matrix(y_test, prediction)

    Sens=mat1[1,1]/mat1[0,1] +mat1[1,1])

    Print(‘sensitivity:’, sens)
```

```
Spec=mat1[0,0]/ (mat1[0,1]+mat1[0,0])

Print('specificity:', spec)
```

Decision Tree Classifier = [823 118 130 154]

Random Forest Classifier= [909 32 138 146]

Support Vector Machine (Linear)=[941 0 284 0]

Support Vector Machine (Rbf)= [881 60 118 166]

Confusion Matrices illustrate true and false predictions. For the Decision tree, we have 823 correctly predicted good wines and 154 correctly predicted bad wines. The same prediction in Random Forest were 909 and 146 respectively. We can easily say that Random Forest is the better method than the decision tree for this white wine data set.

For support vector machine correctly predicted good wine numbers was 941 and correctly predicted bad wines were 0. However, Gaussian kernel these numbers were 881 to 166 respectively. In the and we can say Support vector machine with Gaussian kernel fit the data more.

The last comparison is between Random Forest and Support vector machine rbf. The number of correct predictions under support vector machine rbf is bigger than the random forest. Finally, we can say that support vector machine with radial basis function is better than the Random Forest classifier for this data set.

3.2) SENSITIVITY AND SPECIFICITY

	Decision Tree	Random Forest	SVM_linear	SVM_rbf
Sensitivity	0.542253	0.514084	0.0	0.584507
specificity	0.874601	0.965993	1.0	0.936238

The last step is that checking sensitivity and specificity values. Sensitivity gives us the probability of actual positive cases predicted as positive, On the other hand, specificity is that the actual negatives are predicted as negative. In this study, we assume that sensitivity which means to predict the wine quality accurately is more important than specificity. For this reason, we compare the methods depend on their sensitivity values. According to the table, the highest sensitivity belongs to the Support Vector Machine with radius basis function.

To sum up, we found that the support vector machine with Gaussian Kernel is the best method to predict the white wine samples of Vinho Verde Wines.