Wine Quality Prediction Using IBM & Watson Machine Learning

1. INTRODUCTION

1.1 OVERVIEW

Wine is the most commonly used beverage globally, and its values are considered important in society. Wine is an alcoholic drink that is made up of fermented grapes. Quality of wine is important for its consumers, mainly for producers in the present competitive market to raise the revenue. Wine quality refers to the factors that go into producing a wine, as well as the indicators or characteristics that tell you if the wine is of high quality. Historically, wine quality used to be determined by testing at the end of the production.

If you have come across wine then you will notice that wine has also their type, they are red and white wine. According to experts, wine is differentiated according to its smell, flavour, and colour, but we are not wine experts to say that wine is good or bad. Every person has their own opinion about the tastes, so identifying a quality based on a person's taste is challenging. Judging the quality of wine manually is a really tough task, even the professional wine tasters have the accuracy of 71%.

1.2 PURPOSE

In this project, we present a wine quality prediction technique that utilizes historical data to train simple machine learning models which are more accurate and can help us know the quality of wine. The models can be run on much less resource intensive environments. From this the best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

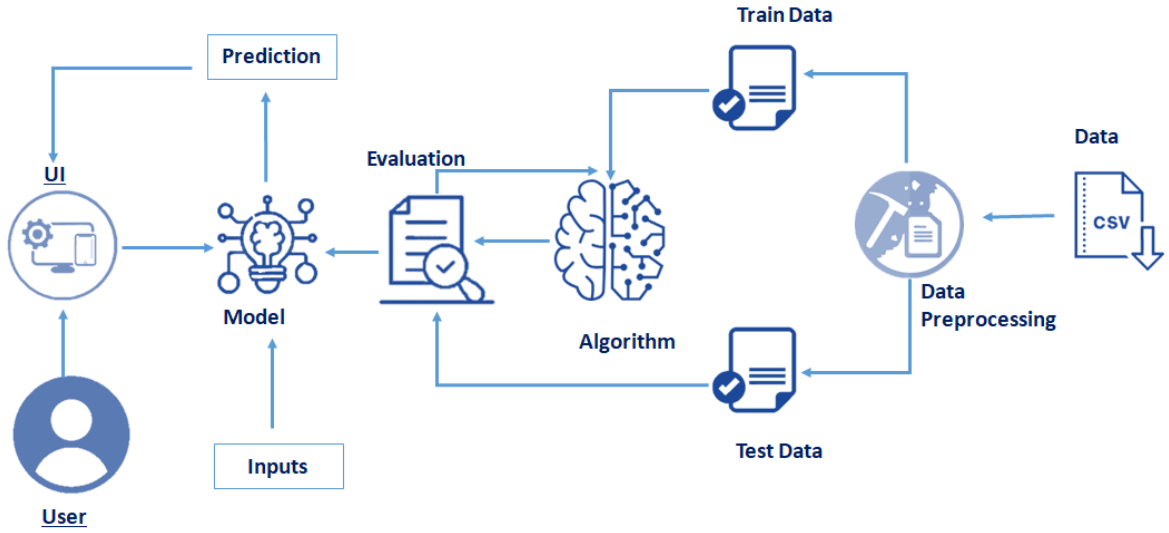
Ensuring the quality of wine whether you are testing for contaminants or developing a new product, we are here to help. Our knowledge, innovative products, and range of solutions allow our customers to maintain their focus where it should be – delivering high-quality wine that consumers expect.

2.2 PROPOSED SYSTEM

Quality of wine is important for its consumers, mainly for producers in the present competitive market to raise the revenue. Wine quality refers to the factors that go into producing a wine, as well as the indicators or characteristics that tell you if the wine is of high quality. Historically, wine quality used to be determined by testing at the end of the production. we present a wine quality prediction technique that utilizes historical data to train simple machine learning models which are more accurate and can help us know the quality of wine. The models can be run on much less resource intensive environments

3. THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE AND SOFTWARE DESIGNING

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was created by Guido van Rossum , and first released on February 20, 1991. Its high-level built in data structures, combined with dynamic typing and dynamic binding , make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Anaconda Navigator

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, crossplatform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder.

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Spyder

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features. Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.Spyder is extensible with first-party and third party plugins includes support for interactive tools for data inspection and embeds Pythonspecific code. Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

Flask

Webframework used for building. It is a web application framework written in python which will be running in local browser with a user interface. In this application, whenever the user interacts with UI and selects emoji, it will suggest the best and top movies of that genre to the user.

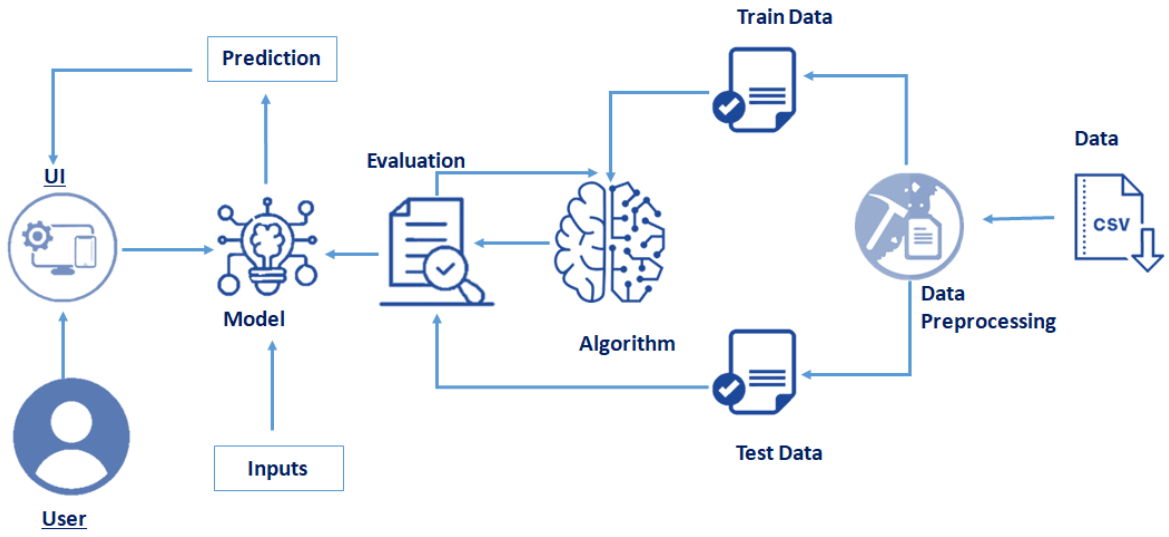
Hardware Requirements:

o Operating system: window 7 and above with 64bit o Processor Type -Intel Core i3-3220 o RAM: 4Gb and above o Hard disk: min 100GB

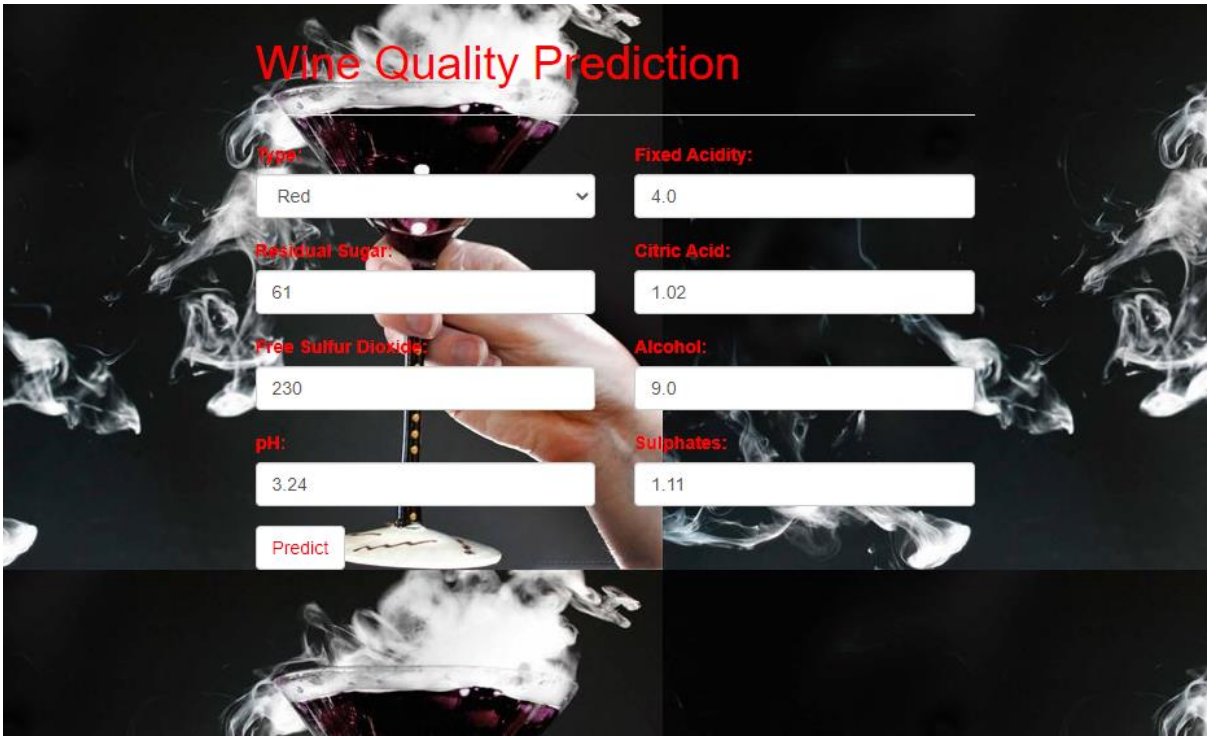
4. EXPERIMENTAL INVESTIGATION

Understanding the demands of wine safety testing can be a complex task for the laboratory with the numerous analytes and residues to monitor.Our separation and detection technologies, combined with experienced applications competence and our best suited chemistries, provide ideal solutions for the analysis of wine. Winemakers know, many of the quality improvements to wine each season, happen in the lab. Regular checks during production and being able to make quick decisions based on the results is key in achieving this quality – therefore, the accuracy and reliability of the data used to make those decisions is critica

5. FLOWCHART



6. RESULT



7. ADVANTAGES

Wine is an alcoholic beverage that is created from grapes (Vitis vinifera). The sugars in grape juice are converted into alcohol during fermentation.
The alcohol (ethanol) in wine blocks various nerve pathways in the brain. Wine also contains antioxidants that might benefit the heart and blood circulation.

DISADVANTAGES

People use wine to prevent heart disease and stroke. It is also used for memory and thinking skills, diabetes, anxiety, cancer and many other purposes, but there's no good scientific evidence to support most of these uses.

8. APPLICATION

Work on several numbers of data: The number of choices for anything on internet is veryhigh and it's tedious to refine most wanted data by self while searching. The scope of this proposal system includes working within numerous data, with case.
Saving of time: Many people have problem selecting the alternative item of movie due to lack of time and due to search issues. Also movie recommendations from friends can be time consuming. The system helps in saving lots of time.

9. CONCLUSION

First, we used oversampling to balance the dataset in the data preprocessing stage to optimize the performance of the model. Then we look for features that can provide better prediction results. For this, we used Pearson coefficient correlation matrices and ranked the features according to the high correlation among the features. After applying the sampling datasets which is balancing dataset the performance of the model is improved. In general, removing irrelevant features of the datasets improved the performance of the classification model. To conclude that the minority classes of a dataset will not get a good representation on a classifier and representation for each class can be solved by oversampling and undersampling to balance the representation classes over datasets. Therefore, in the classification algorithms by selecting the appropriate features and balancing the data can improve the performance of the model.

10. FUTURE SCOPE

In this project, we present a wine quality prediction technique that utilizes historical data to train simple machine learning models which are more accurate and can help us know the quality of wine. The models can be run on much less resource intensive environments. From this the best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

11. BIBLIOGRAPHY

https://www.kaggle.com/code/vishalyo990/prediction-of-quality-of-wine

APPENDIX

Installing Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.svm import SVC
import pickle

**Importing the Dataset**

data = pd.read_csv(r'C:\Users\eisha\Documents\python\Wine Quality
Prediction\Dataset\winequalityN.csv')
data.head()

data.columns

data.describe()

data.info()

data.isnull().sum()

data['quality'].value_counts()

**Data Visualization**

Univariate Analysis

# Checking the distribution (normal or skewed)

plt.figure(figsize=(12,5))
sns.distplot(data['alcohol'],color='r')
plt.show()

df_cat = data.select_dtypes(include='object')
df_cat.head()

# Visualizing the count of categorical variable.

plt.figure(figsize=(20,5))
for i,j in enumerate(df_cat):
    plt.subplot(1,4,i+1)
    sns.countplot(data[j])

axarr = data.hist(column=['quality'], bins=100, figsize=(6, 6))
ax = axarr.flatten()[0]
ax.set_xlabel(f"{ax.get_title()} value")
ax.set_ylabel("Quantity")
title = ax.get_title()
ax.set_title(f"Histogram of {title}")
plt.show()
```

Bivariate Analysis

Visualizing the relation between quality and type

```
plt.figure(figsize=(10,5))
sns.countplot(data['quality'],hue=data['type'])
plt.legend(loc='upper right')
```

checking the variation of fixed acidity in the different qualities of wine

```
plt.scatter(data['quality'], data['fixed acidity'], color = 'green')
plt.title('relation of fixed acidity with wine')
plt.xlabel('quality')
plt.ylabel('fixed acidity')
plt.legend()
plt.show()
```

checking the variation of fixed acidity in the different qualities of wine

```
plt.bar(data['quality'], data['alcohol'], color = 'maroon')
plt.title('relation of alcohol with wine')
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.legend()
plt.show()
```

Composition of citric acid go higher as we go higher in the quality of the wine

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid', data = data)
```

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = data)
```

#Composition of chloride also go down as we go higher in the quality of the wine

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = data)
```

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = data)
```

#Sulphates level goes higher with the quality of wine

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = data)
```

#Sulphates level goes higher with the quality of wine

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = data)
```

As we can see that like the above two items do not have very strong relation to the dependent variable we have to showcase a correlation plot to check which of the items are more related to the dependent variable and which items are less related to the dependent variables.

```
f, ax = plt.subplots(figsize=(10, 8))
corr = data.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10,
as_cmap=True),
        square=True, ax=ax)

plt.figure(figsize = (20, 10))
sns.heatmap(data.corr().abs(), annot = True)
plt.show()
```

From the above correlation plot for the given dataset for wine quality prediction, we can easily see which items are related strongly with each other and which items are related weekly with each other. For Example,
The strongly correlated items are :

1.fixed acidity and citric acid.
2.free sulphur dioxide and total sulphor dioxide.
3.fixed acidity and density.
4. alcohol and quality.

so, from above points there is a clear inference that alcohol is the most important characteristic to determine the quality of wine.

The weekly correlated items are :

1.citric acid and volatile acidity.
2.fixed acidity and ph.
3.density and alcohol.

These are some relations which do not depend on each other at all.

#Descriptive Analysis
`data.describe()`

Data pre-processing

```
# Removing Unnecassary columns from the dataset
# As we saw that volatile acidity, total sulphor dioxide, chlorides, density are very less related to the dependent variable
#   quality so even if we remove these columns the accuracy won't be affected that much.

data = data.drop(['volatile acidity', 'total sulfur dioxide', 'chlorides', 'density'], axis = 1)
```

```
# checking the shape of the dataset
print(data.shape)

data.columns

# converting the response variables(3-7) as binary response variables that is either good or bad
data['quality'] = data['quality'].map({3 : 'bad', 4 :'bad', 5: 'bad',
                                       6: 'good', 7: 'good', 8: 'good'})

# analyzing the different values present in the dependent variable(quality column)
data['quality'].value_counts()

data['type'].value_counts()

data.isnull().any()

data.isnull().sum()

data["fixed acidity"].fillna(data["fixed acidity"].mean(),inplace = True)
data["sulphates"].fillna(data["sulphates"].mean(),inplace = True)
data["pH"].fillna(data["pH"].mean(),inplace = True)
data["residual sugar"].fillna(data["residual sugar"].mean(),inplace = True)
data["citric acid"].fillna(data["citric acid"].mean(),inplace = True)
data["quality"].fillna(data["quality"].mode()[0],inplace = True)

data.isnull().any()



#converting categorical data to numerical data
le = LabelEncoder()
data['quality'] = le.fit_transform(data['quality'])
data['type'] = le.fit_transform(data['type'])



sns.countplot(data['quality'])

# dividing the dataset into dependent and independent variables
x = data.iloc[:,:8]
y = data.iloc[:,8:9]
# determining the shape of x and y.
print(x.shape)
print(y.shape)

# dividing the dataset in training and testing set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 44)

# determining the shapes of training and testing sets
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
# standard scaling
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

Modelling

Logistic Regression

```
# creating the model
model = LogisticRegression()
# feeding the training set into the model
model.fit(x_train, y_train)
# predicting the results for the test set
y_pred = model.predict(x_test)
# calculating the training and testing accuracies
print("Training accuracy :", model.score(x_train, y_train))
print("Testing accuracy :", model.score(x_test, y_test))
# classification report
print(classification_report(y_test, y_pred))
# confusion matrix
print(confusion_matrix(y_test, y_pred))
```

Stochastic Gradient Descent Classifier

```
# creating the model
model = SGDClassifier(penalty=None)

# feeding the training model into the model
model.fit(x_train, y_train)

# predicting the values for the test set
y_pred = model.predict(x_test)

# classification report
print(classification_report(y_test, y_pred))
```

Support Vector Machine

```
# creating the model
model = SVC()

# feeding the training set into the model
model.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(x_train, y_train))
```

```
print("Testing accuracy :", model.score(x_test, y_test))


# finding the best parameters for the SVC model

param = {
    'C': [0.8,0.9,1,1.1,1.2,1.3,1.4],
    'kernel':['linear', 'rbf'],
    'gamma' :[0.1,0.8,0.9,1,1.1,1.2,1.3,1.4]
}
grid_svc = GridSearchCV(model, param_grid = param, scoring = 'accuracy', cv = 10)


#grid_svc.fit(x_train, y_train)

#grid_svc.best_params_

# creating a new SVC model with these best parameters

model2 = SVC(C = 1.4, gamma = 0.1, kernel = 'rbf')
model2.fit(x_train, y_train)
y_pred = model2.predict(x_test)

print(classification_report(y_test, y_pred))

**Decision Forest**

# creating model
model = DecisionTreeClassifier()

# feeding the training set into the model
model.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(x_train, y_train))
print("Testing accuracy :", model.score(x_test, y_test))

# classification report
print(classification_report(y_test, y_pred))

# confusion matrix
print(confusion_matrix(y_test, y_pred))


#Now lets try to do some evaluation for decision tree model using cross validation.

model_eval = cross_val_score(estimator = model, X = x_train, y = y_train, cv = 10)
model_eval.mean()
```

Random Forest

```
# creating the model
rfmodel = RandomForestClassifier(n_estimators = 200)

# feeding the training set into the model
rfmodel.fit(x_train, y_train)

# predicting the results for the test set
y_pred = rfmodel.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", rfmodel.score(x_train, y_train))
print("Testing accuracy :", rfmodel.score(x_test, y_test))

# Random Forest classification report
classification_report(y_test, y_pred)

# Random Forest confusion matrix
confusion_matrix(y_test, y_pred)

#Random Forest Cross validation score
model_eval = cross_val_score(estimator = rfmodel, X = x_train, y = y_train, cv = 5)
model_eval.mean()

def logisticRegression(x_train, x_test, y_train, y_test):
    # creating the model
    model = LogisticRegression()
    # feeding the training set into the model
    model.fit(x_train, y_train)
    # predicting the results for the test set
    y_pred = model.predict(x_test)
    # calculating the training and testing accuracies
    print('***logisticRegression***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))

def SGD(x_train, x_test, y_train, y_test):
    # creating the model
    model = SGDClassifier(penalty=None)
    # feeding the training model into the model
    model.fit(x_train, y_train)
    # predicting the values for the test set
    y_pred = model.predict(x_test)
    print('***Stochastic Gradient Descent Classifier***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
```

```
    print(confusion_matrix(y_test, y_pred))


def SVClassifier(x_train, x_test, y_train, y_test):
    # creating the model
    model = SVC()
    # feeding the training set into the model
    model.fit(x_train, y_train)
    # predicting the results for the test set
    y_pred = model.predict(x_test)
    # calculating the training and testing accuracies
    print('***Support Vector Classifier***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))

def decisionTree(x_train, x_test, y_train, y_test):
    dt=DecisionTreeClassifier()
    dt.fit(x_train,y_train)
    yPred = dt.predict(x_test)
    print('***DecisionTreeClassifier***')
    print("Training accuracy :", dt.score(x_train, y_train))
    print("Testing accuracy :", dt.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))

def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print("Training accuracy :", rf.score(x_train, y_train))
    print("Testing accuracy :", rf.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))

def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print('***GradientBoostingClassifier***')
    print("Training accuracy :", xg.score(x_train, y_train))
    print("Testing accuracy :", xg.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

```
def compareModel(x_train, x_test, y_train, y_test):
    logisticRegression(x_train, x_test, y_train, y_test)
    print('-'*100)
    SGD(x_train, x_test, y_train, y_test)
    print('-'*100)
    SVClassifier(x_train, x_test, y_train, y_test)
    print('-'*100)
    decisionTree(x_train, x_test, y_train, y_test)
    print('-'*100)
    randomForest(x_train, x_test, y_train, y_test)
    print('-'*100)
    xgboost(x_train, x_test, y_train, y_test)
    print('-'*100)

compareModel(x_train, x_test, y_train, y_test)

pickle.dump(rfmodel,open('wineQuality_new.pkl','wb'))
```