1) **Recreate the "output" tab from the data_table.xlsx using Pandas (and with Numpy if required)**
   a. **There are formulas in the "pivot" tab (E6:G5928) so you can see how the fields are calculated**
   b. **Use %%time at the top of your code so that it measures the speed of code execution**
   c. **Provide the .describe() output of the table**
   d. **Provide a screenshot of your code with the timing**

```
# Importing Pandas, time, and uploading raw data
%load_ext autotime
import pandas as pd
data = pd.read_excel(r'data_table.xlsx', sheet_name='data')
```

```
# Making a dataframe to represent as the pivot table
df = pd.DataFrame({'Row Labels':data['Name'],
        'RiskType':data['RiskType'],
        'ACG':data.Value[data['Factor']=='ACG'],
        'KEI':data.Value[data['Factor']=='KEI'],
        'UTE':data.Value[data['Factor']=='UTE']} )
# Sorting dataframe columns by the "Row Label" column
df = df.sort_values(by=['Row Labels'], inplace=False, ascending=True)
# Selecting rows that the risk is equivalent to 1 and filling NaN values with 0
df=df[df['RiskType']==1].fillna(0)
del df['RiskType']
# Combining similar 'Row Labels' into one. In pivot table, similar labels are combined together.
# In this code, we filled NaN values with 0, we can sum similar labels and get the same result.
df2 = df.groupby(['Row Labels'], as_index=False)['ACG',"KEI",'UTE'].sum()
df2.rename(columns={'Row Labels':'Name'}, inplace=True)
# Calculating and adding Field1, Field2, and Field3 to the dataframe
# This calculation were done by SUMIFS() function in Excel.
df2['Field1']=df2[  (df2['KEI']>0) ]['ACG'].sum()-df2['ACG']
df2['Field2']=df2[  (df2['ACG']>0) ]['KEI'].sum()-df2['KEI']
df2['Field3']=df2[  (df2['ACG']>0) ]['UTE'].sum()-df2['UTE']
# The output is the same as 'output' tab of the row-data with converting floats to integers
# and replacing 0 values with no data
df2=df2.astype(int, errors='ignore')
df2.replace(0, '', inplace=True)
df2.to_csv('OUTPUT.xlsx', index=False)
```

```
time: 267 ms
```

```
df2.head()
```

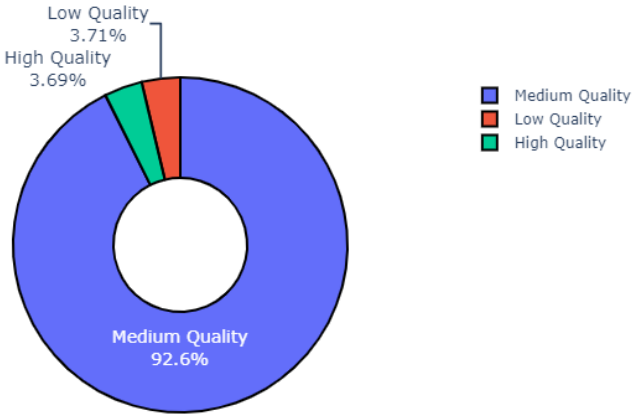
	Name	ACG	KEI	UTE	Field1	Field2	Field3
0	AADRDK		2939984		71219373	68827211	76246040
1	AADSRZ		-1416188		71219373	73183383	76246040
2	AAKQUC		9924728		71219373	61842467	76246040
3	AALDYP		1897540		71219373	69869655	76246040
4	AAOKUV		6536697		71219373	65230498	76246040

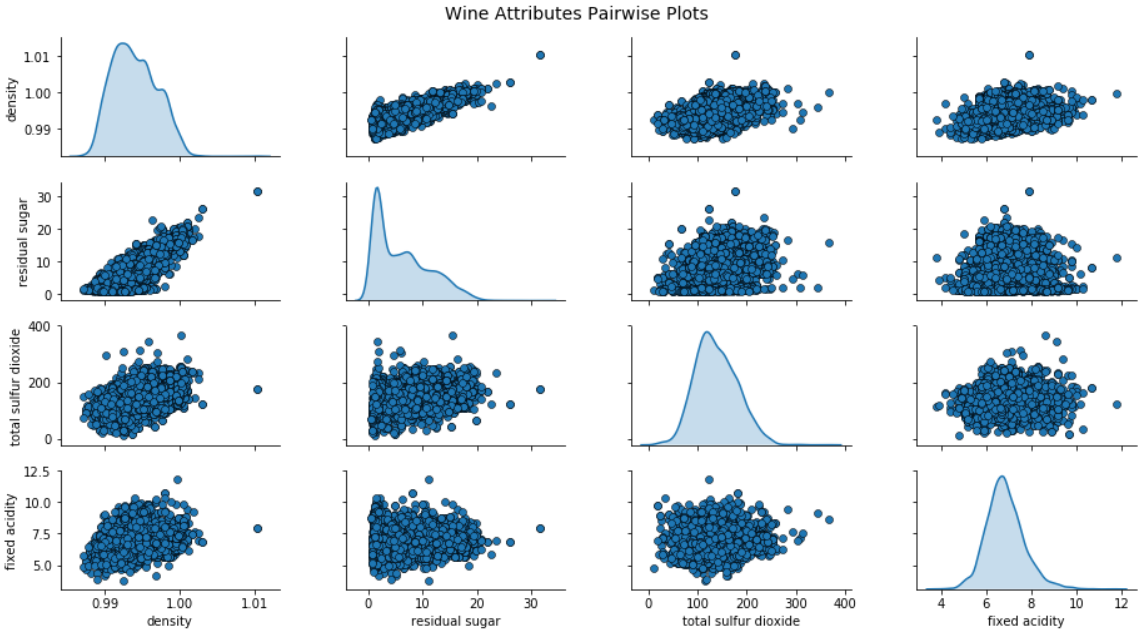
```
df2.describe()
```

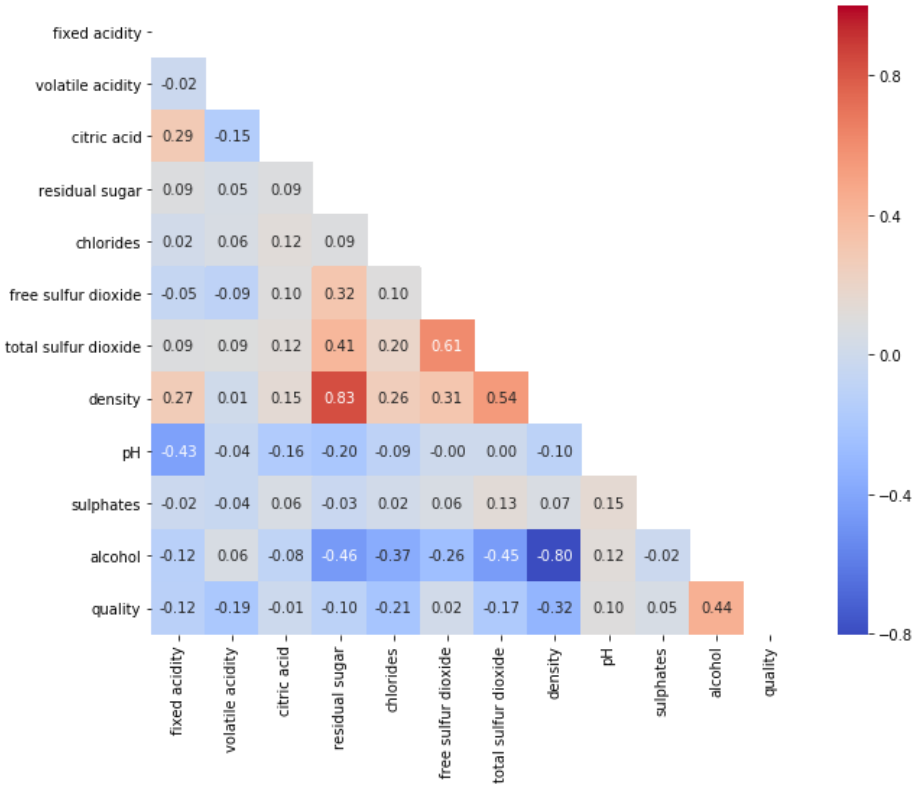
	KEI	Field1	Field2	Field3
count	5.923000e+03	5.923000e+03	5.923000e+03	5.923000e+03
mean	3.531468e+06	7.120460e+07	6.823573e+07	7.580902e+07
std	4.938089e+06	4.384447e+05	4.938089e+06	2.055686e+06
min	-4.999352e+06	5.962278e+07	5.976735e+07	6.426672e+07
25%	-7.363655e+05	7.121937e+07	6.399905e+07	7.624604e+07
50%	3.619486e+06	7.121937e+07	6.814771e+07	7.624604e+07
75%	7.768142e+06	7.121937e+07	7.250356e+07	7.624604e+07
max	1.199985e+07	7.608136e+07	7.676655e+07	8.123847e+07

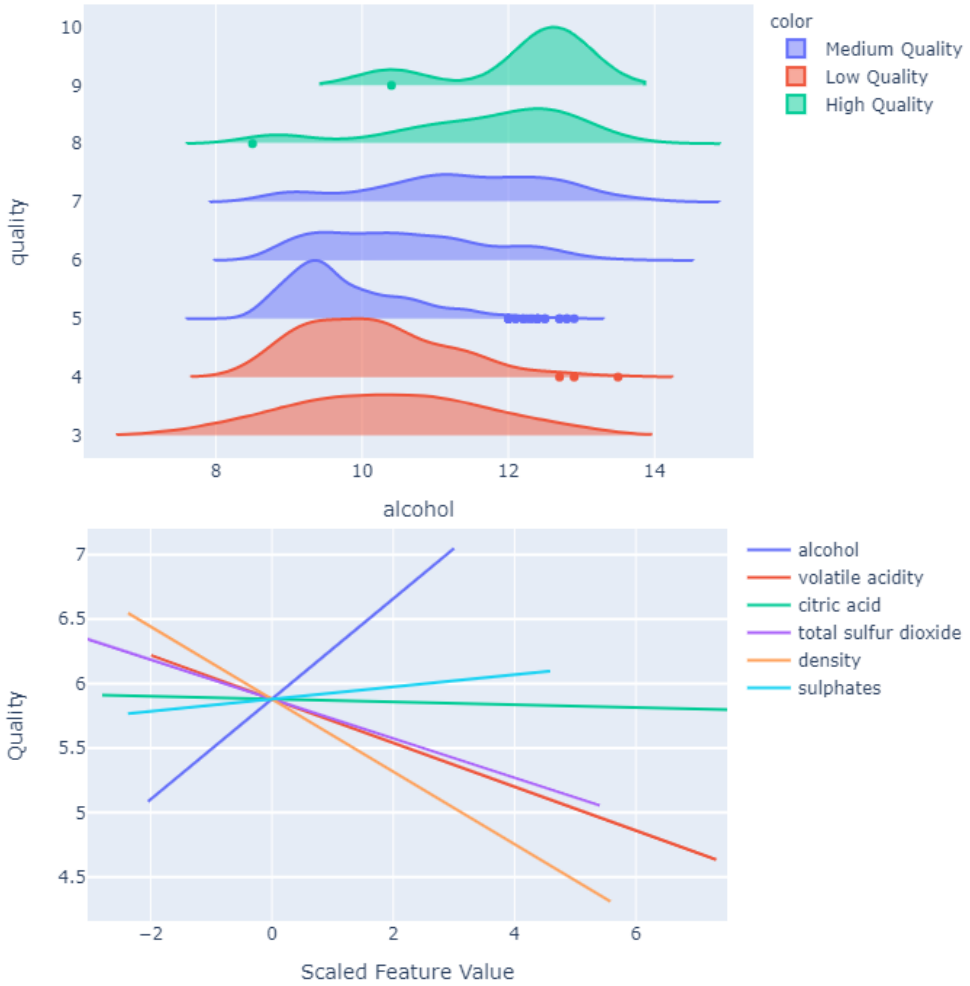
---

2) **Find three (3) interesting insights / patterns in the wine_data.csv**
a. **Provide screenshots of at least two graphs of the patterns using plotly**


Low Quality
3.71%
High Quality
3.69%
Medium Quality
92.6%
Medium Quality
Low Quality
High Quality


Wine Attributes Pairwise Plots
density
residual sugar
total sulfur dioxide
fixed acidity


fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality
0.8
0.4
0.0
−0.4
−0.8



b. Explain why the insights/patterns are interesting

The followings are the findings that we can extract from the previous figures:

1- Having more residual sugar in wine has a direct co-relationship with wine density.
More Residual Sugar = Higher Density

2- Having more sulfur dioxide in wine has a direct co-relationship with density.
More Total Sulfur Dioxide = Higher Density

Based on No.1 and 2, we should expect to have this relationship:

3- Having more residual sugar in wine has a direct co-relationship with the total sulfur dioxide.
More Residual Sugar = Higher Total Sulfur Dioxide

Now, it is interesting to know what are the effects of these features on wine quality based on the correlation matrix:

4- Density, total sulfur dioxide, and residual sugar have an inverse co-relationship with quality.
More Sulfur Dioxide = Lower Quality
More Density = Lower Quality
More Residual Sugar = Lower Quality

The question that comes into mind is to know what is the reason behind these relationships.

A quick search about Sulfur Dioxide shows that it is used as an anti-oxidant, and it is beneficial for inhibiting the development of “bad germs” in the wine, but it also prevents the natural aromas of the grapes and having high amount of it leads to headaches and allergies. So, **the first interesting insight is that having Sulfur Dioxide decreases quality based on its bad effects on wine aroma, and negative health effects. This is the first reason for us to choose high-quality wine that has less amount of sulfur dioxide.**

The next question that comes into mind is to find out how residual sugar and sulfur dioxide have the same negative effect on wine quality.

As an example of how much sulfur dioxide is allowed in wine, one of the strictest regulations, the European one (Reg. CE No 606/2009) establishes the following limits:

Dry Reds: 150 mg/L (150 ppm)
Dry Whites and Rosés: 200 mg/L (200 ppm)
Sweet reds (more than 5 g/L of sugars): 200 mg/L (200 ppm)
Sweet Whites and Rosés (more than 5 g/L of sugars): 250 mg/L (250 ppm)

Thus, the sweeter the wine is, it has more sulfur dioxide. Thus, sweet wines have more Sulfur Dioxide and they considered as low-quality wine. **This gives us the second interesting insight: we can recognize high-quality wine by its sweetness, and it would be better to buy dry wines to have less sulfur dioxide.**

Another question, is alcohol percentage a good metric for the wine quality? These are things we can find out by analyzing the relationship between features:

5- Having more alcohol has an inverse co-relationship with density
More alcohol = Lower density
6- Having more alcohol has an inverse co-relationship with residual sugar
More alcohol = Lower residual sugar

Based on No.5 and 6, we should expect to have this relationship:

7- Having more alcohol has a co-relationship with quality
More alcohol = Higher quality

This is the formula for calculating wine density:

Alcohol + Sugar + Water= Density

Based on the second insight, it would be better if we choose a wine with lower residual sugar, and by assuming fixed amount of water and considering previous formula, we will reach into conclusion that it would be better to drink wine with higher alcohol to have a higher quality wine with less density. **Thus, the third insight is that we should prefer wine with a higher amount of alcohol to have less sweet and low-density wine.**

To sum, we can represent three tips based on our data and the relations between features to buy a high-quality wine:

- Choosing wine with less sulfur dioxide
- Choosing dry wine over sweet ones
- Choosing wine with a higher percentage of alcohol

c. Provide a screenshot of your code

The code I wrote for this task is not just for extracting graphs, and it contains some of machine learning algorithms like RandomForest, DecisionTree, GridSearch, and SMOTE. In this part, I will only show the code screenshots related to the graphs, and other parts of the code for data prediction will be available after task 3 in this report.

```
df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	4.7	0.785	0.00	3.4	0.036	23.0	134.0	0.98981	3.53	0.92	13.8	6
1	9.2	0.710	0.23	6.2	0.042	15.0	93.0	0.99480	2.89	0.34	10.1	6
2	6.4	0.690	0.09	7.6	0.044	34.0	144.0	0.99480	3.26	0.38	10.1	6
3	6.9	0.270	0.49	23.5	0.057	59.0	235.0	1.00240	2.98	0.47	8.6	5
4	5.9	0.220	0.45	22.6	0.120	55.0	122.0	0.99636	3.10	0.35	12.8	5

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4882 entries, 0 to 4881
Data columns (total 12 columns):
fixed acidity           4882 non-null float64
volatile acidity        4882 non-null float64
citric acid             4882 non-null float64
residual sugar          4882 non-null float64
chlorides               4882 non-null float64
free sulfur dioxide     4882 non-null float64
total sulfur dioxide    4882 non-null float64
density                 4882 non-null float64
pH                      4882 non-null float64
sulphates               4882 non-null float64
alcohol                 4882 non-null float64
quality                 4882 non-null int64
dtypes: float64(11), int64(1)
```

In this code, the quality of wines is categorized by an integer between 0 and 10 in this data, and we would like to categorize this range of integers to something more sensible. So, we will categorize wines to the three categories of Low Quality (quality<5), Medium Quality (5<=quality<=7), and High Quality (quality>7).

```
quality = df["quality"].values
category = []
for num in quality:
    if num<5:
        category.append("Low Quality")
    elif num>7:
        category.append("High Quality")
    else:
        category.append("Medium Quality")
category = pd.DataFrame(data=category, columns=["category"])
data = pd.concat([df,category],axis=1)
data.drop(columns="quality",axis=1,inplace=True)
```

Code for making pie plot:

```
colors = ['gold', 'mediumturquoise', 'darkorange']
fig = go.Figure(data=[go.Pie(labels=data['category'].value_counts().index,
                             values=data['category'].value_counts().values, hole=.4)])
fig.update_traces(hoverinfo='value', textinfo='label+percent', textfont_size=14,
                  insidetextorientation='horizontal'
                  ,marker=dict( line=dict(color='#000000', width=2)))
fig.show()
data["category"].value_counts()
```

Code for making violin plot:

```
px.violin(df,y='quality',x='alcohol', color=data['category'],
          orientation='h').update_traces(side='positive',width=2)
```

Code for making correlation map:

```
corr = df.corr()
fig, ax = plt.subplots(figsize=(10, 8))
mask = np.triu(np.ones_like(corr, dtype=np.bool))
sns.heatmap(corr,mask=mask, cmap='coolwarm', annot=True, fmt=".2f")
plt.show()
```

Code for making the plot with the slope of each feature:

```
scaler = StandardScaler()
X = df.drop('quality', axis=1)
X = scaler.fit_transform(X)
new_df = pd.DataFrame(X, columns=df.drop('quality',axis=1).columns)
new_df['quality'] = df['quality']
new_df.head()
```

```
# alcohol
slope_alcohol, intercept_alcohol, r_value, p_value, std_err = stats.linregress(new_df['alcohol'],
                                                                               new_df['quality'])
# volatile acidity
slope_volatile, intercept_volatile, r_value, p_value, std_err = stats.linregress(new_df['volatile acid
                                                                                 new_df['quality'])
# citric acid
slope_citric, intercept_citric, r_value, p_value, std_err = stats.linregress(new_df['citric acid'],
                                                                             new_df['quality'])
# total sulfur dioxide
slope_sulfur, intercept_sulfur, r_value, p_value, std_err = stats.linregress(new_df['total sulfur diox
                                                                             new_df['quality'])
# density
slope_density, intercept_density, r_value, p_value, std_err = stats.linregress(new_df['density'],
                                                                               new_df['quality'])
# sulphates
slope_sulphates, intercept_sulphates, r_value, p_value, std_err = stats.linregress(new_df['sulphates']
                                                                                   new_df['quality'])
```

```
alcohol = go.Scatter(x=new_df['alcohol'],
                     y=slope_alcohol*new_df['alcohol']+intercept_alcohol,
                     name='alcohol')
volatile = go.Scatter(x=new_df['volatile acidity'],
                      y=slope_volatile*new_df['volatile acidity']+intercept_volatile,
                     name='volatile acidity')
citric = go.Scatter(x=new_df['citric acid'],
                      y=slope_citric*new_df['citric acid']+intercept_citric,
                     name='citric acid')
sulfur = go.Scatter(x=new_df['total sulfur dioxide'],
                      y=slope_sulfur*new_df['total sulfur dioxide']+intercept_sulfur,
                     name='total sulfur dioxide')
density = go.Scatter(x=new_df['density'],
                      y=slope_density*new_df['density']+intercept_density,
                     name='density')
sulphates = go.Scatter(x=new_df['sulphates'],
                      y=slope_sulphates*new_df['sulphates']+intercept_sulphates,
                     name='sulphates')
layout = dict(title = 'Relationshop between features and quality',
              yaxis = dict(
                  title='Quality',
                  zeroline=False,
                  gridwidth=2
              ),
              xaxis = dict(zeroline = False,title='Scaled Feature Value')
             )
datas = [alcohol, volatile, citric, sulfur, density, sulphates]
fig = go.Figure(data=datas, layout=layout)
py.iplot(fig)
```

Code for making scatter plots:

```
# Pair-wise Scatter Plots
cols = ['density', 'residual sugar', 'total sulfur dioxide', 'fixed acidity']
pp = sns.pairplot(df[cols], size=1.8, aspect=1.8,
                  plot_kws=dict(edgecolor="k", linewidth=0.5),
                  diag_kind="kde", diag_kws=dict(shade=True))

fig = pp.fig
fig.subplots_adjust(top=0.93, wspace=0.3)
t = fig.suptitle('Wine Attributes Pairwise Plots', fontsize=14)
```

---

3. ****BONUS** --- not required**
   a. **Provide a screenshot of a Tensorflow 2 code that would compile a model to use 4 GPUs simultaneously**

The following code can use 4 GPUs for computing matrix multiplication and using a single CPU to perform an element-wise sum over the matrices. In this code, each GPU is feed with a different batch of data, which is handled by towers, and each GPU is responsible for a batch of data.

```
import tensorflow as tf

c = []
a = tf.get_variable(f"a", [2, 2, 3], initializer=tf.random_uniform_initializer(-1, 1))
b = tf.get_variable(f"b", [2, 3, 2], initializer=tf.random_uniform_initializer(-1, 1))

# Multiple towers
for i, d in enumerate(['/gpu:0', '/gpu:1','/gpu:2', '/gpu:3']):
    with tf.device(d):
        c.append(tf.matmul(a[i], b[i]))   # Tower i is responsible for batch data i.

with tf.device('/cpu:0'):
    sum = tf.add_n(c)

sess = tf.Session(config=tf.ConfigProto(log_device_placement=True, allow_soft_placement=True))

init = tf.global_variables_initializer()
sess.run(init)

print(sess.run(sum))
```

Appendix: Wine Quality Prediction

```
X = data.drop('category', axis = 1)
y = data['category']
```

```
#Encoding wine categories. We have three categories and this will change the categories to 0,1, and 2.
labelencoder_y =LabelEncoder()
y= labelencoder_y.fit_transform(y)
```

```
#Spliting dataset to training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,random_state=1)
```

```
#Scaling the data
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
print('X_train Shape:',X_train.shape ,'\nX_test Shape:',X_test.shape,'\ny_train Shape:',y_train.shape ,
```

```
X_train Shape: (3905, 11)
X_test Shape: (977, 11)
y_train Shape: (3905,)
y_test Shape: (977,)
```

```
#As we saw in the pie chart at EDA section, data enteries are not categorized uniformly and this is
#not acceptable.SMOTE method is used for generating data of categories with fewer data enteries and
#solving imbalance data problem0.
for _ in range(len(np.unique(y_train))-1):
    smote=SMOTE("minority")
    X_train,y_train=smote.fit_sample(X_train,y_train)
print("y_train per category after balancing data:",Counter(y_train))
```

```
y_train per category: Counter({2: 3623, 1: 3623, 0: 3623})
```

```
#As we saw in the pie chart at EDA section, data enteries are not categorized uniformly and this is
#not acceptable.SMOTE method is used for generating data of categories with fewer data enteries and
#solving imbalance data problem0.
for _ in range(len(np.unique(y_train))-1):
    smote=SMOTE("minority")
    X_train,y_train=smote.fit_sample(X_train,y_train)
print("y_train per category after balancing data:",Counter(y_train))
```

```
y_train per category: Counter({2: 3623, 1: 3623, 0: 3623})
```

```
#Prediction of wine quality by Support Vector Machine algorithm
svc = SVC()
svc.fit(X_train,y_train)
pred_svc =svc.predict(X_test)
print(classification_report(y_test,pred_svc))
print('Confusion Matrix:\n',confusion_matrix(y_test, pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.16      0.72      0.26        46
           1       0.18      0.64      0.28        33
           2       0.96      0.70      0.81       898

    accuracy                           0.69       977
   macro avg       0.43      0.68      0.45       977
weighted avg       0.90      0.69      0.76       977

Confusion Matrix:
 [[ 33   1  12]
 [  0  21  12]
 [179  94 625]]
```

```
#Prediction of wine quality by K-Nearest Neighbor algorithm
knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
pred_knn=knn.predict(X_test)
print(classification_report(y_test, pred_knn))
print('Confusion Matrix:\n',confusion_matrix(y_test, pred_knn))
```

```
              precision    recall  f1-score   support

           0       0.21      0.63      0.31        46
           1       0.17      0.55      0.26        33
           2       0.96      0.78      0.86       898

    accuracy                           0.77       977
   macro avg       0.45      0.65      0.48       977
weighted avg       0.90      0.77      0.82       977

Confusion Matrix:
 [[ 29   0  17]
 [  2  18  13]
 [110  85 703]]
```

```
#Prediction of wine quality by Random Forest algorithm
rfc = RandomForestClassifier(n_estimators=250)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
print(classification_report(y_test, pred_rfc))
print('Confusion Matrix:\n',confusion_matrix(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.44      0.52      0.48        46
           1       0.45      0.45      0.45        33
           2       0.96      0.95      0.95       898

    accuracy                           0.91       977
   macro avg       0.62      0.64      0.63       977
weighted avg       0.92      0.91      0.91       977

Confusion Matrix:
 [[ 24   1  21]
 [  0  15  18]
 [ 30  17 851]]
```

```
# feature importance
importances = pd.DataFrame({'feature':['fixed acidity', 'volatile acidity', 'citric acid', 'residual s
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol'],
                            'importance':np.round(rfc.feature_importances_,3)})
importances = importances.sort_values('importance',ascending=False).set_index('feature')
importances
```

feature	importance
alcohol	0.165
free sulfur dioxide	0.150
volatile acidity	0.113
chlorides	0.087
density	0.084
fixed acidity	0.077
total sulfur dioxide	0.077
residual sugar	0.071
citric acid	0.064
pH	0.058
sulphates	0.054

```
#Finding best parameters for our Random Forest model with GridSearch method

param = {
    'n_estimators': [700,900,1100],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [17,19,21,23],
    'criterion' :['gini', 'entropy']
}

grid_rfc = GridSearchCV(rfc, param_grid=param, scoring='accuracy', cv=10)
```

```
grid_rfc.fit(X_train, y_train)
```

```
GridSearchCV(cv=10, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=250, n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [17, 19, 21, 23],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [700, 900, 1100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

```
grid_rfc.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 23,
 'max_features': 'log2',
 'n_estimators': 1100}
```

```
#Let's run our RFC again with the best parameters.
rfc2 = RandomForestClassifier(n_estimators = 1100, max_features =  'log2', max_depth= 23,
                              criterion= 'entropy')
rfc2.fit(X_train, y_train)
pred_rfc2 = rfc2.predict(X_test)
print(classification_report(y_test, pred_rfc2))
```

```
              precision    recall  f1-score   support

           0       0.44      0.48      0.46        46
           1       0.45      0.45      0.45        33
           2       0.95      0.95      0.95       898

    accuracy                           0.91       977
   macro avg       0.62      0.63      0.62       977
weighted avg       0.91      0.91      0.91       977
```

Conclusion:

In this project, we analyzed data for predicting wine quality and for this purpose, we trained different machine learning models. Random Forest method provided the best results. This method could predict wine quality (0.91% accuracy in analyzed metrics) better than other methods with cosidering different metrics such ass precision, recall and F1-score.