In this project we will MANUALLY use basic and fundamental methods of machine learning with help of linear algebra such as:

```
1) Linear Regression with least-squares error
2) PCA with Singular value decomposition (SVD)
3) Logistic Regression
4) Support Vector Machines (SVM)
```

Tasks:

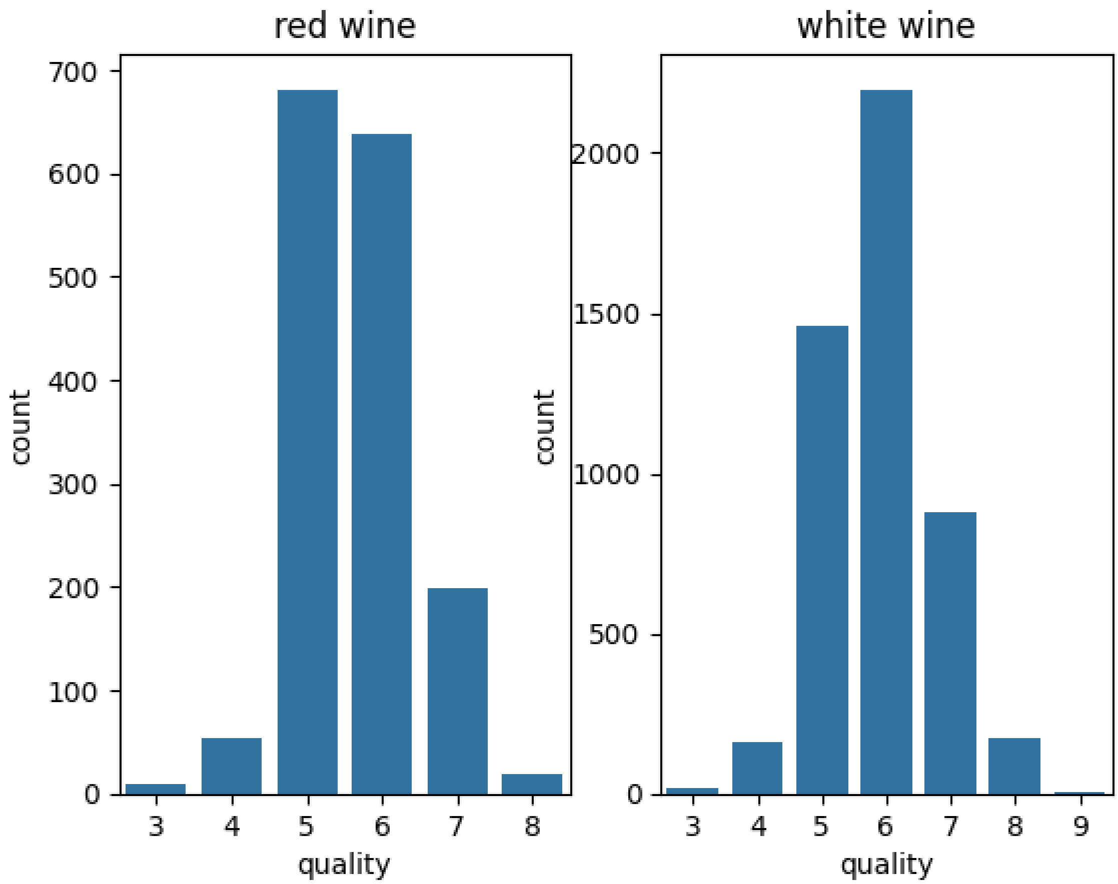
```
1) Predicting wine quality
2) Dimensionality reduction
3) Wine classification
```

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

red_wine = pd.read_csv('winequality-red.csv', sep=';')
white_wine = pd.read_csv('winequality-white.csv', sep=';')

display(red_wine.head(5))
display(white_wine.head(5))
```

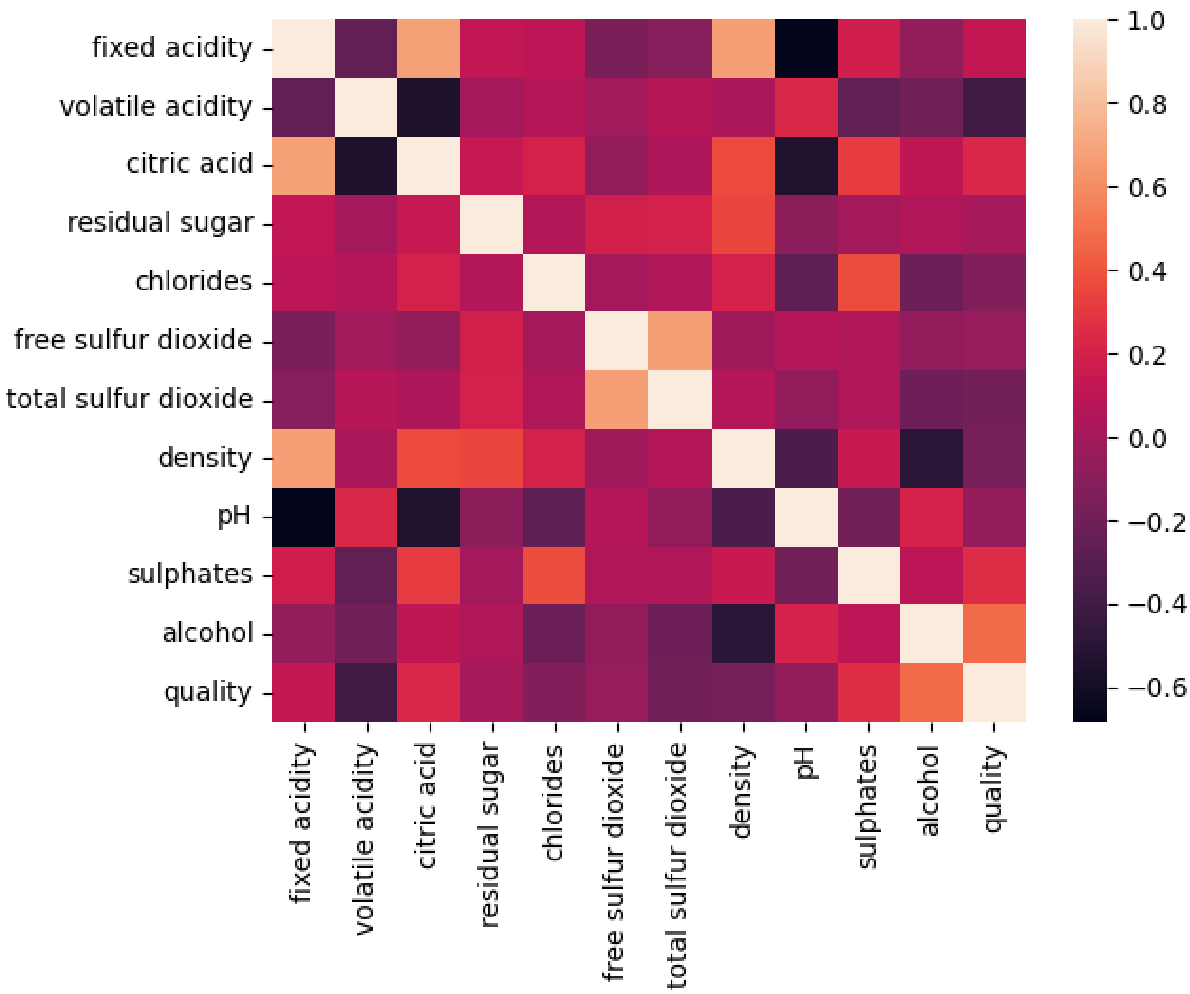
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	

In [2]:
```
fig, ax = plt.subplots(nrows=1,ncols=2)
sns.countplot(data=red_wine, x='quality', ax=ax[0])
sns.countplot(data=white_wine, x='quality', ax=ax[1])

ax[0].set_title('red wine')
ax[1].set_title('white wine')

plt.show()
```



In [3]:
```
sns.heatmap(data=red_wine.corr())
plt.show()
```



Steps we will apply:

```
1) Data normalization
2) PCA for feature selection
3) Least-square method to predict wine quality
4) Rate model by MSE score
```

In [4]:
```
#Data normalization

norm_func = lambda x: x - x.mean()
x_red_wine = red_wine.apply(norm_func)

x_red_wine.head(5)
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
0	-0.919637	0.172179	-0.270976	-0.638806	-0.011467	-4.874922	-12.467792	0.001053	(
1	-0.519637	0.352179	-0.270976	0.061194	0.010533	9.125078	20.532208	0.000053	-(
2	-0.519637	0.232179	-0.230976	-0.238806	0.004533	-0.874922	7.532208	0.000253	-(
3	2.880363	-0.247821	0.289024	-0.638806	-0.012467	1.125078	13.532208	0.001253	-(
4	-0.919637	0.172179	-0.270976	-0.638806	-0.011467	-4.874922	-12.467792	0.001053	(

In [5]:

```
#Creating training and testing datasets

half_size = int(len(x_red_wine)/2)

y_red_train = red_wine['quality'][:half_size]
y_red_test = red_wine['quality'][half_size:]

x_red_wine_train = x_red_wine.loc[:,'fixed acidity':'alcohol'][:half_size]
x_red_wine_test = x_red_wine.loc[:,'fixed acidity':'alcohol'][half_size:]

display(x_red_wine_train.head(3))
display(x_red_wine_test.head(3))

display(y_red_train.head(3))
display(y_red_test.head(3))
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
0	-0.919637	0.172179	-0.270976	-0.638806	-0.011467	-4.874922	-12.467792	0.001053	0.19
1	-0.519637	0.352179	-0.270976	0.061194	0.010533	9.125078	20.532208	0.000053	-0.11
2	-0.519637	0.232179	-0.230976	-0.238806	0.004533	-0.874922	7.532208	0.000253	-0.05

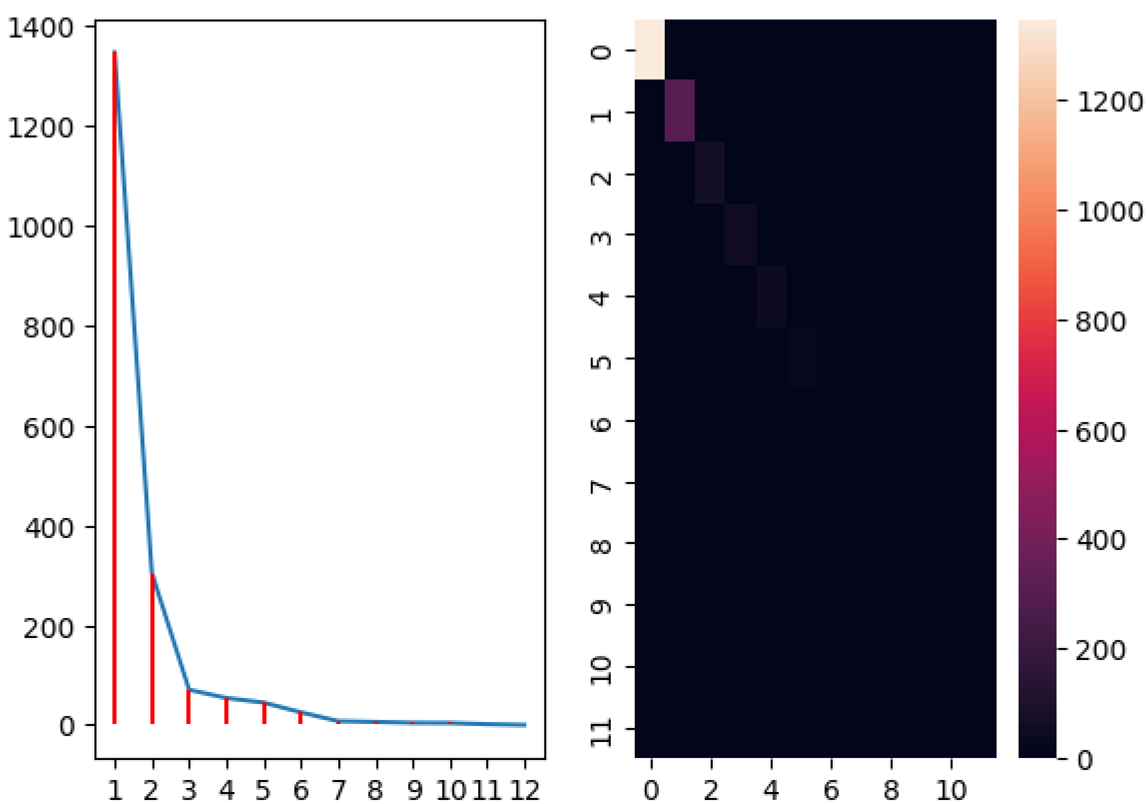
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
799	1.080363	-0.027821	0.069024	1.061194	-0.005467	-10.874922	-32.467792	0.001953
800	-1.119637	0.082179	-0.190976	1.461194	-0.005467	10.125078	61.532208	-0.000337
801	0.280363	0.022179	-0.180976	0.761194	-0.019467	-7.874922	-29.467792	0.000603

```
0    5
1    5
2    5
Name: quality, dtype: int64
```

```
799    6
800    5
801    5
Name: quality, dtype: int64
```

In [6]:

```
#PCA for feature selection

lsv, sv, rsv = np.linalg.svd(x_red_wine)

diag_sv = np.diag(sv)

fig, ax = plt.subplots(nrows=1,ncols=2)

sns.heatmap(data=diag_sv)
sns.lineplot(x=list(range(1, len(sv)+1)), y = sv, ax = ax[0])
ax[0].set_xticks(list(range(1, len(sv)+1)))

for i in range(1, len(sv)):
    ax[0].vlines(i, ymin=0, ymax=sv[i-1], color='red')

plt.show()
```



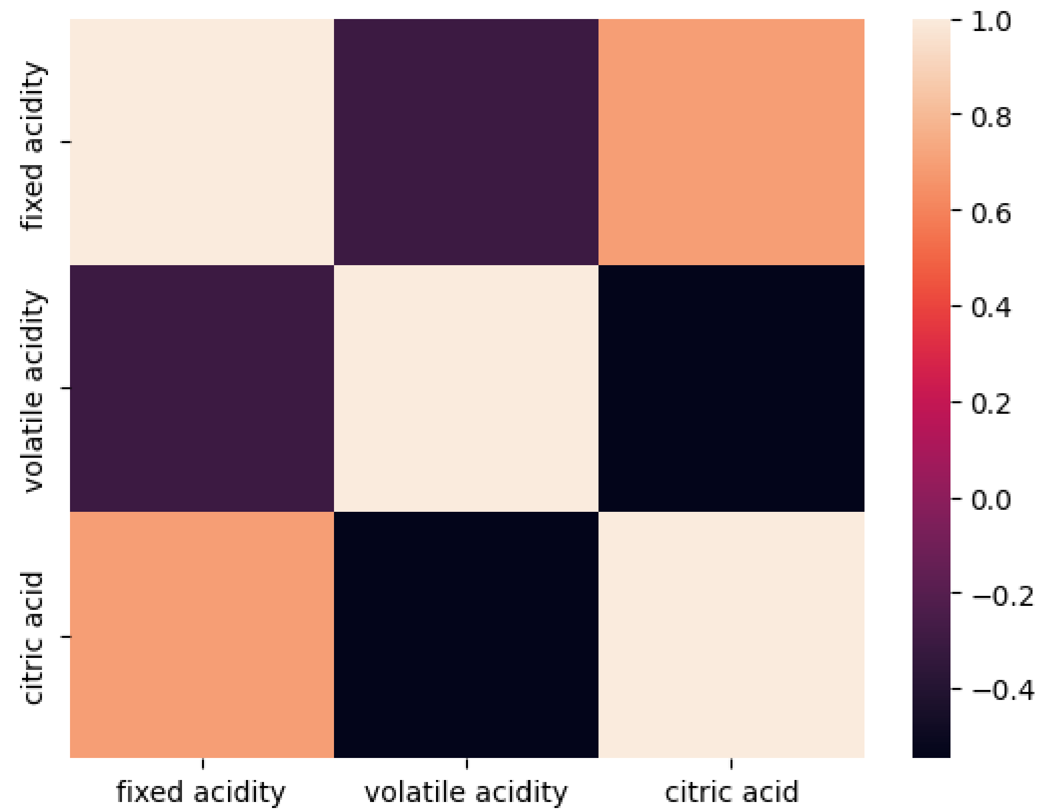
We can see that we can use 3 components for our analysis and reduce them

In [7]:

```
#Reduction

x_red_reduced_train = x_red_wine_train.iloc[:,:3]
x_red_reduced_test = x_red_wine_test.iloc[:,:3]
display(x_red_reduced_train.head(5))
```

```
sns.heatmap(x_red_reduced_train.corr())
plt.show()
```

	fixed acidity	volatile acidity	citric acid
0	-0.919637	0.172179	-0.270976
1	-0.519637	0.352179	-0.270976
2	-0.519637	0.232179	-0.230976
3	2.880363	-0.247821	0.289024
4	-0.919637	0.172179	-0.270976



Least-square method to predict wine quality

$$Q = \beta + \sum_{i=1}^{n} X_i \beta_i$$

where Q - wine quality, X_i - wine's feature, β_i - regression coefficient, β - regression constant

$$Ax = Q$$

A - features matrix extended with ones in left (for β regression constant), x - regression coefficients vector, Q = quality vector

$$x = A^{-1}Q$$

In [8]:
```
#Least-square method to predict wine quality

#Adding b-const column
b_ones = np.ones(x_red_reduced_train.shape[0]).T
A_red = np.column_stack((b_ones, x_red_reduced_train))

#using pseudo invers because we do not have square matrix
x = np.linalg.pinv(A_red) @ y_red_train
print(x)
```

```
[ 5.52785152  0.06867977 -1.28993431 -0.22361438]
```

These regression coefficient represents our formula to find(predict) wine quality by its 3 components

$$Q = 5.52785152 + 0.06867977x_1 - 1.28993431x_2 - 0.22361438x_3$$

In [9]:
```
coefs = x

def predict_quality(features):
    Q = coefs[0]
    i = 1
    while i - 1 < len(features):
        Q += coefs[i]*features[i-1]
        i += 1
    return Q
```

In [10]:
```
features = [-0.919637, 0.172179, -0.270976]
predicted_quality = round(predict_quality(features))

print('data:', x_red_reduced_train.head(1))
print('quality: ', y_red_train.head(1)[0])
print('predicted quality: ', predicted_quality)
```

```
data:     fixed acidity  volatile acidity  citric acid
0      -0.919637          0.172179    -0.270976
quality:  5
predicted quality:  5
```

We will rate our model with MSE, less value we've got - better

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2.$$

In [11]:
```
# Rate model with MSE score
pred_num = len(x_red_wine_test)
mse = 0

i = 0
while i < pred_num:
    features = x_red_reduced_test.iloc[i].to_numpy()
    real_quality = y_red_test.iloc[i]

    predicted_quality = int(predict_quality(features))
```

```
    mse += np.power(real_quality - predicted_quality,2)

    i += 1


mse /= pred_num
print(mse)
```

```
1.2325
```

We have MSE score 1.2325 what is not so bad with respect that we used simple least-square error model, let's check what it would be without dimensionality reduction:

In [12]:
```
x_red_reduced_train = x_red_wine_train
x_red_reduced_test = x_red_wine_test

b_ones = np.ones(x_red_reduced_train.shape[0]).T
A_red = np.column_stack((b_ones, x_red_reduced_train))

x = np.linalg.pinv(A_red) @ y_red_train
coefs = x

def predict_quality(features):
    Q = coefs[0]
    i = 1
    while i - 1 < len(features):
        Q += coefs[i]*features[i-1]
        i += 1
    return Q

pred_num = len(x_red_wine_test)
mse = 0

i = 0
while i < pred_num:
    features = x_red_reduced_test.iloc[i].to_numpy()
    real_quality = y_red_test.iloc[i]

    predicted_quality = int(predict_quality(features))

    mse += np.power(real_quality - predicted_quality,2)

    i += 1


mse /= pred_num
print(mse)
```

```
0.7025
```

Less a bit, but better

In [ ]:
```

```