Predicting wine quality using machine learning algorithms

Juraj Todić

Osijek, 2024

Contents

Introduction....3

Description of the dataset used....4

Data pre-processing....5

Statistical analysis of pure wines....9

Training a model for predicting the grade of 100% Graševina....13

1. Neural network....16

2. Random Forest algorithm....18

3. SVR....19

4. XGBoost (Extreme Gradient Boosting)....20

Results....21

Conclusion....23

Literature....24

Introduction

The task was to create a model with the ability to predict the quality of wine based on the physical and chemical properties of the wine.

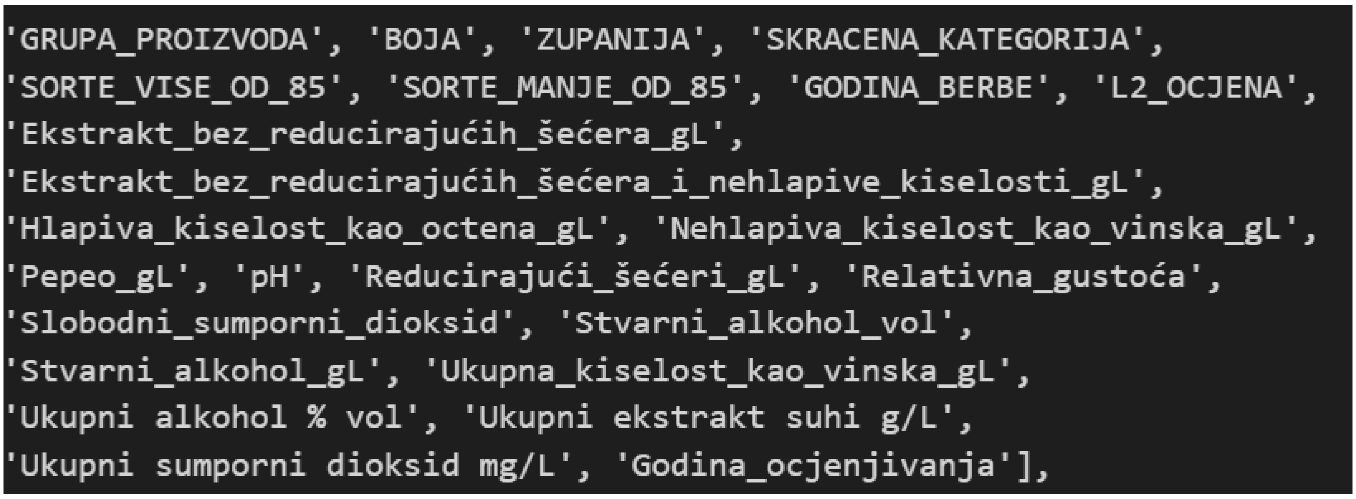
Wine is the most widely used beverage globally, and its value is considered important in society. The quality of wine is always important for its consumers, and especially for producers in today's competitive market to increase revenues. Historically, the quality of wine was determined by testing at the end of production; Up to that stage, a lot of time and money is already spent. If the quality is not good, it is necessary to apply various procedures from the very beginning, which is very expensive. Each person has their own opinion about taste, so it's challenging to determine quality based on personal taste. With the development of technology, manufacturers began to use various devices for testing in the development stages. This way they can get a better idea of the quality of the wine, which of course saves a lot of money and time. In addition, this helped to collect a large amount of data with different parameters, such as the amount of different chemicals and the temperature used during production and the quality of the wine produced.

Analysis of the basic parameters that determine the quality of wine is crucial. Machine learning can be an alternative to identify the most important parameters that control the quality of wine. In this paper, we have shown how machine learning can be used to predict the quality of wine itself.

An extensive dataset of 42833 examples was used, which contained physicochemical properties and appropriate quality. We combined these datasets, built and trained several models. We evaluated their performance and chose the best model. Wine quality assessments are modeled as a regression problem.

Description of the dataset used

The dataset that was used was obtained in .xlsx format and consists of 42833 rows. The data is divided into several excel sheets, where each represents the year when the readings were taken, the years when the readings were collected are 2015-2021. All sheets are merged into one dataFrame and a "Godina_ocjenjivanja" column has been added. In the following image we can see all the columns in the data set:



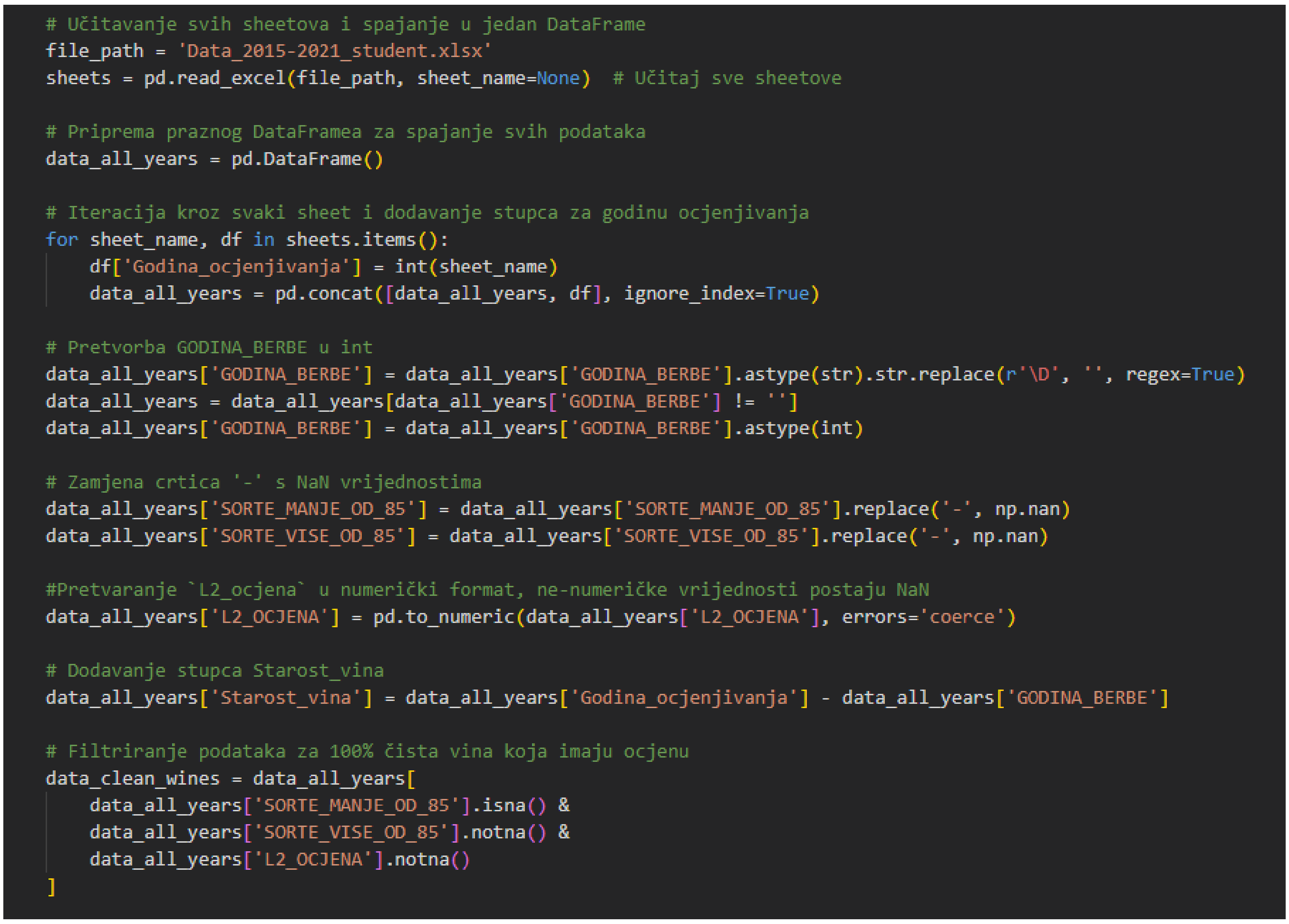
```
'GRUPA_PROIZVODA', 'BOJA', 'ZUPANIJA', 'SKRACENA_KATEGORIJA',
'SORTE_VISE_OD_85', 'SORTE_MANJE_OD_85', 'GODINA_BERBE', 'L2_OCJENA',
'Ekstrakt_bez_reducirajućih_šećera_gL',
'Ekstrakt_bez_reducirajućih_šećera_i_nehlapive_kiselosti_gL',
'Hlapiva_kiselost_kao_octena_gL', 'Nehlapiva_kiselost_kao_vinska_gL',
'Pepeo_gL', 'pH', 'Reducirajući_šećeri_gL', 'Relativna_gustoća',
'Slobodni_sumporni_dioksid', 'Stvarni_alkohol_vol',
'Stvarni_alkohol_gL', 'Ukupna_kiselost_kao_vinska_gL',
'Ukupni alkohol % vol', 'Ukupni ekstrakt suhi g/L',
'Ukupni sumporni dioksid mg/L', 'Godina_ocjenjivanja'],
```

Figure 1 – Columns in the dataset

The dataset contains general information about what type of wine it is, information about the purity of wine, its origin, grade, and a number of chemical and physical parameters.

Data pre-processing

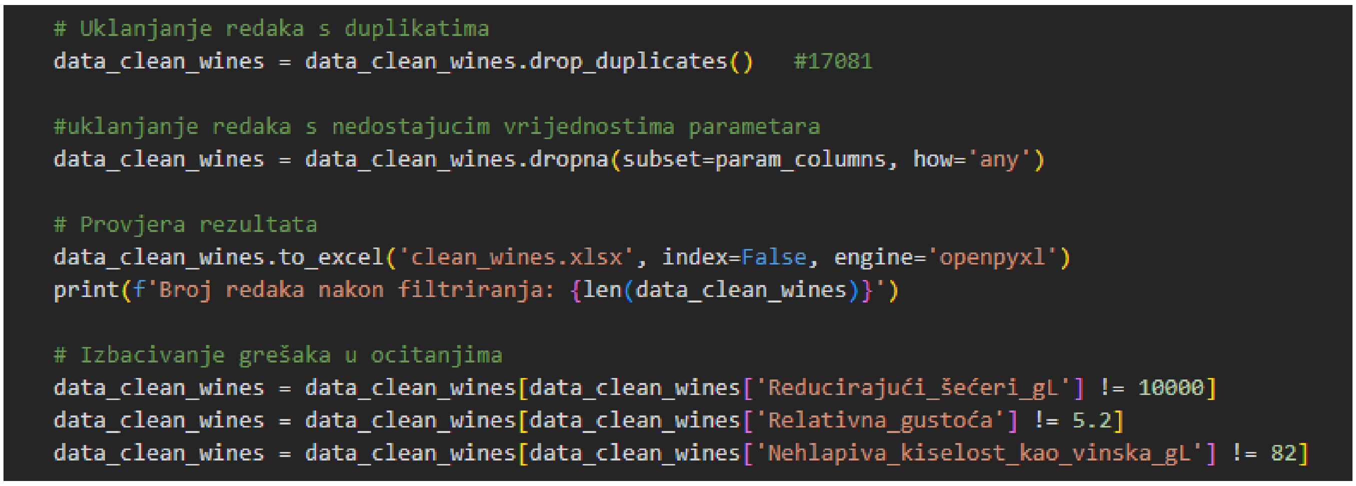
In order to be able to use the data in model training, we first performed data preprocessing. We have converted the values from "GODINA_BERBE" to int, we have replaced the blank data label from the hyphen character (-) to the NaN value, we have also converted L2_ocjenu to numeric format. In order to define new parameters that can have a correlation with the grade, we have added a soup "Starost_vina" as the difference between the year of evaluation and the year of harvest. In order to get clear results and give a good profile to the types of wines, we filtered the data for only 100% pure wines, without looking at wine blends. We have also removed rows that do not have a value of "L2_ocjena" or have some missing value of chemical parameters. After that, we removed the rows that are duplicates, that is, they have repeating values.



```python
# Učitavanje svih sheetova i spajanje u jedan DataFrame
file_path = 'Data_2015-2021_student.xlsx'
sheets = pd.read_excel(file_path, sheet_name=None)  # Učitaj sve sheetove

# Priprema praznog DataFramea za spajanje svih podataka
data_all_years = pd.DataFrame()

# Iteracija kroz svaki sheet i dodavanje stupca za godinu ocjenjivanja
for sheet_name, df in sheets.items():
    df['Godina_ocjenjivanja'] = int(sheet_name)
    data_all_years = pd.concat([data_all_years, df], ignore_index=True)

# Pretvorba GODINA_BERBE u int
data_all_years['GODINA_BERBE'] = data_all_years['GODINA_BERBE'].astype(str).str.replace(r'\D', '', regex=True)
data_all_years = data_all_years[data_all_years['GODINA_BERBE'] != '']
data_all_years['GODINA_BERBE'] = data_all_years['GODINA_BERBE'].astype(int)

# Zamjena crtica '-' s NaN vrijednostima
data_all_years['SORTE_MANJE_OD_85'] = data_all_years['SORTE_MANJE_OD_85'].replace('-', np.nan)
data_all_years['SORTE_VISE_OD_85'] = data_all_years['SORTE_VISE_OD_85'].replace('-', np.nan)

#Pretvaranje `L2_ocjena` u numerički format, ne-numeričke vrijednosti postaju NaN
data_all_years['L2_OCJENA'] = pd.to_numeric(data_all_years['L2_OCJENA'], errors='coerce')

# Dodavanje stupca Starost_vina
data_all_years['Starost_vina'] = data_all_years['Godina_ocjenjivanja'] - data_all_years['GODINA_BERBE']

# Filtriranje podataka za 100% čista vina koja imaju ocjenu
data_clean_wines = data_all_years[
    data_all_years['SORTE_MANJE_OD_85'].isna() &
    data_all_years['SORTE_VISE_OD_85'].notna() &
    data_all_years['L2_OCJENA'].notna()
]
```

Figure 2 – Preprocessing code

After that, using the box plot, we displayed all the values of the parameters of pure wines, and found that we had several errors in data entry/reading and several outliers that we should remove from the data set. Ejection of outliers in only necessary conditions, with the aim of minimizing data loss.



```python
# Uklanjanje redaka s duplikatima
data_clean_wines = data_clean_wines.drop_duplicates()   #17081

#uklanjanje redaka s nedostajucim vrijednostima parametara
data_clean_wines = data_clean_wines.dropna(subset=param_columns, how='any')

# Provjera rezultata
data_clean_wines.to_excel('clean_wines.xlsx', index=False, engine='openpyxl')
print(f'Broj redaka nakon filtriranja: {len(data_clean_wines)}')

# Izbacivanje grešaka u ocitanjima
data_clean_wines = data_clean_wines[data_clean_wines['Reducirajući_šećeri_gL'] != 10000]
data_clean_wines = data_clean_wines[data_clean_wines['Relativna_gustoća'] != 5.2]
data_clean_wines = data_clean_wines[data_clean_wines['Nehlapiva_kiselost_kao_vinska_gL'] != 82]
```

Figure 3 – Data filtering

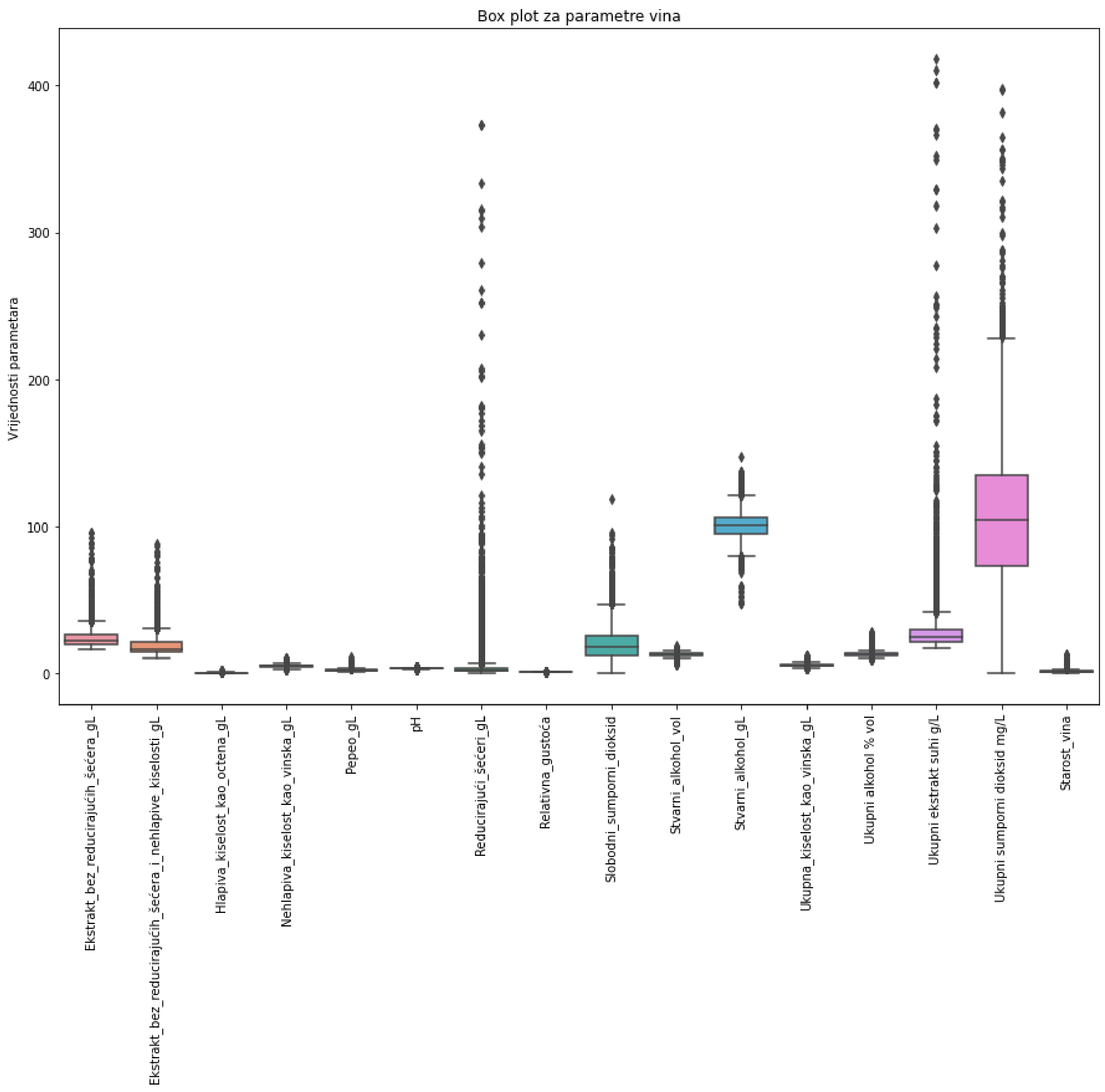
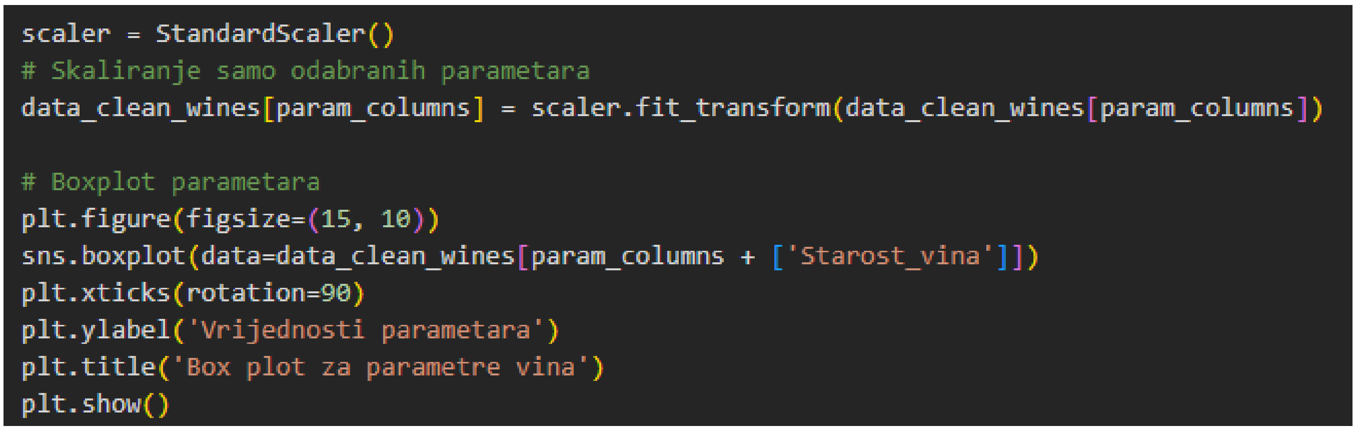


Figure 4 – Box plot parameters of pure wines

It is quite common for datasets to contain unbalanced and unnormalized data, the distributions of wine characteristics have different scales, so it will be convenient to normalize them.



```
scaler = StandardScaler()
# Skaliranje samo odabranih parametara
data_clean_wines[param_columns] = scaler.fit_transform(data_clean_wines[param_columns])

# Boxplot parametara
plt.figure(figsize=(15, 10))
sns.boxplot(data=data_clean_wines[param_columns + ['Starost_vina']])
plt.xticks(rotation=90)
plt.ylabel('Vrijednosti parametara')
plt.title('Box plot za parametre vina')
plt.show()
```

Figure 5 – Data normalization

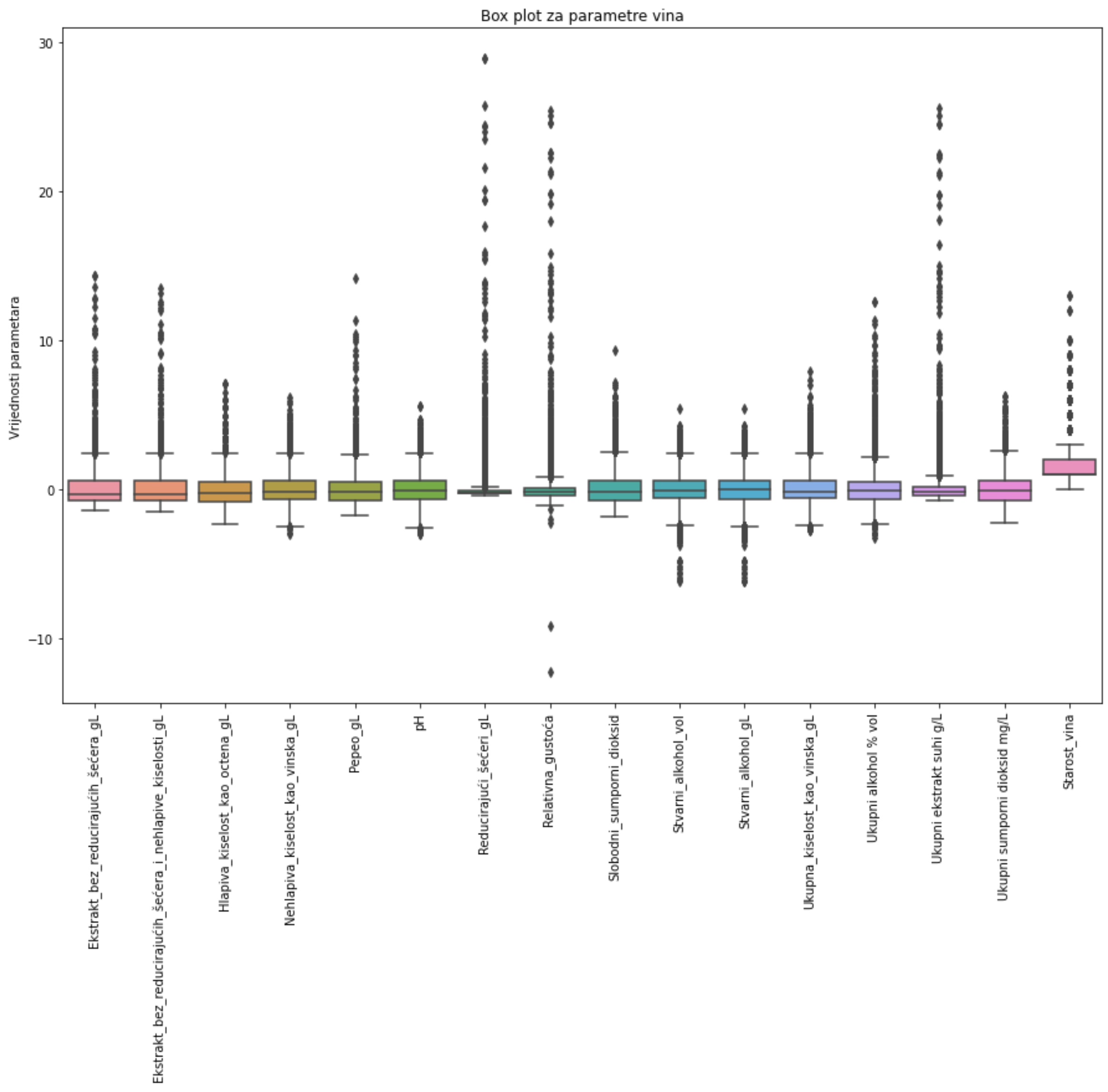


Figure 6 – Box plot of normalized parameters of pure wines

Statistical analysis of pure wines

Next, we'll take a look at some interesting and relevant statistics related to our dataset.

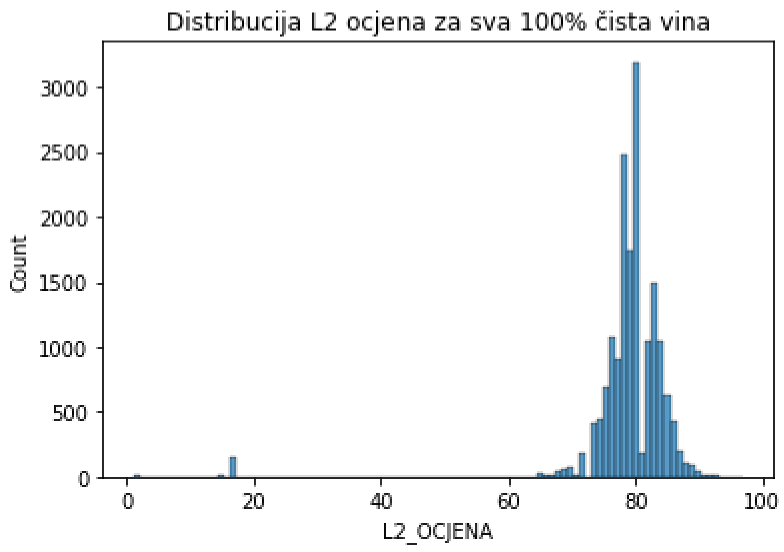


Figure 7 – Distribution of wine ratings bar graf

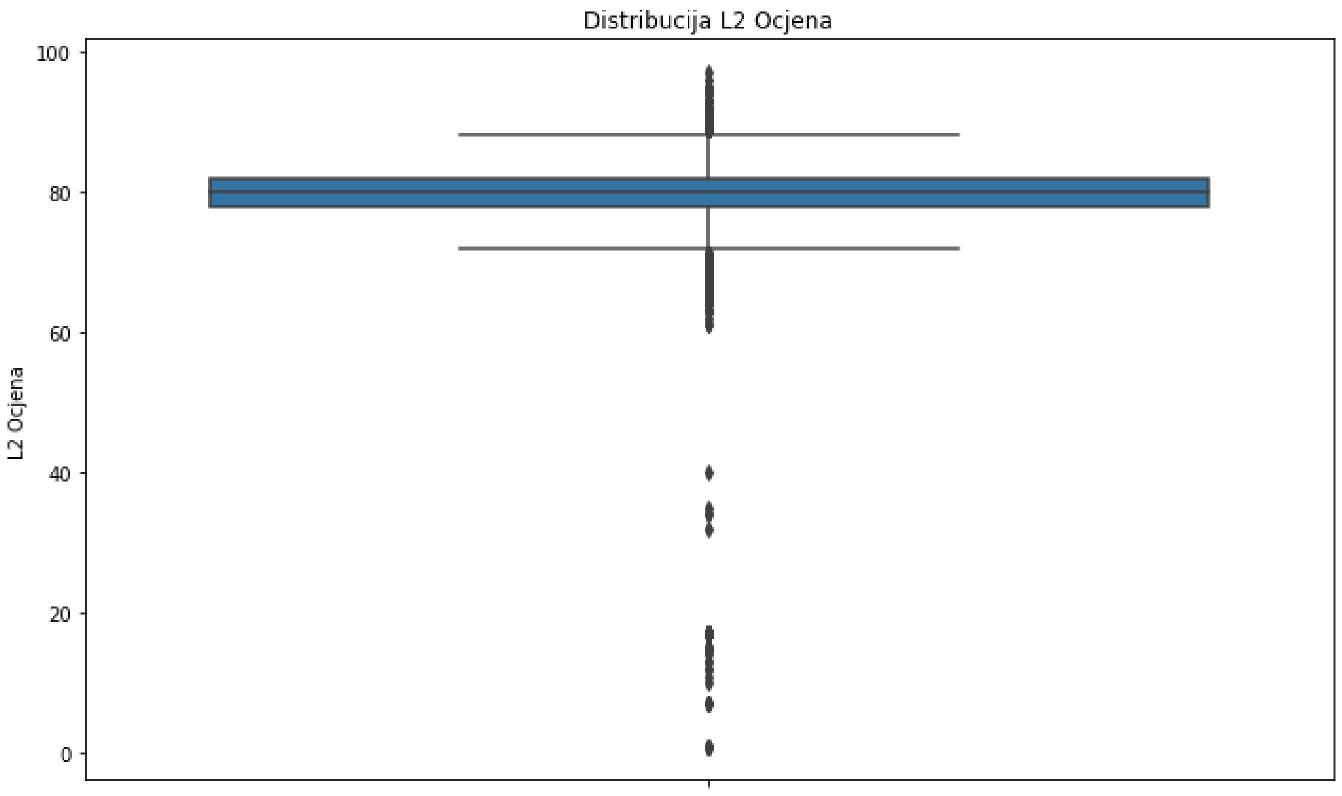


Figure 8 – Distribution of box plot wine ratings

We see that the vast majority of wines are rated with a score in the 70 – 90 segment, distorted data can cause the model to be biased towards the majority class or values, ignoring minority cases, resulting in poor performance on underrepresented data. In other words, with skewed data, the model can overadjust to most or prevailing trends in data, learning patterns that cannot be well generalized to more balanced or diverse datasets. This problem can otherwise be solved by using *resampling* techniques, e.g. minority class oversampling, majority class subsampling. It is also interesting to note the distribution, i.e. lack of rating values in the 20-30 and 40-60 segments.

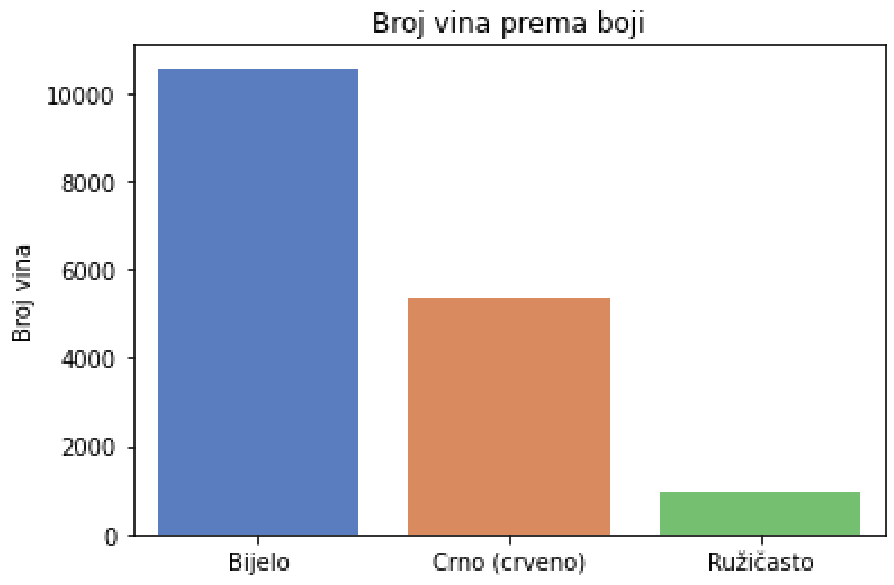


Figure 9 – Number of wines by colour

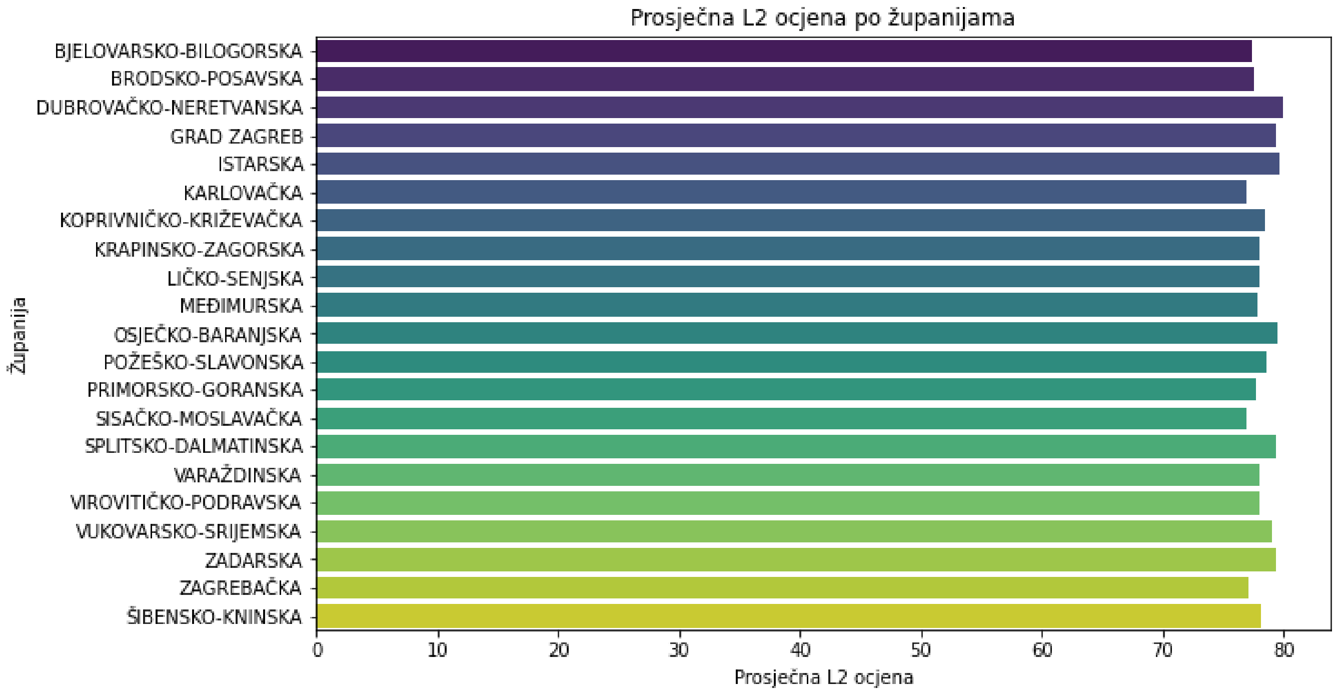


Figure 10 – Average score by county

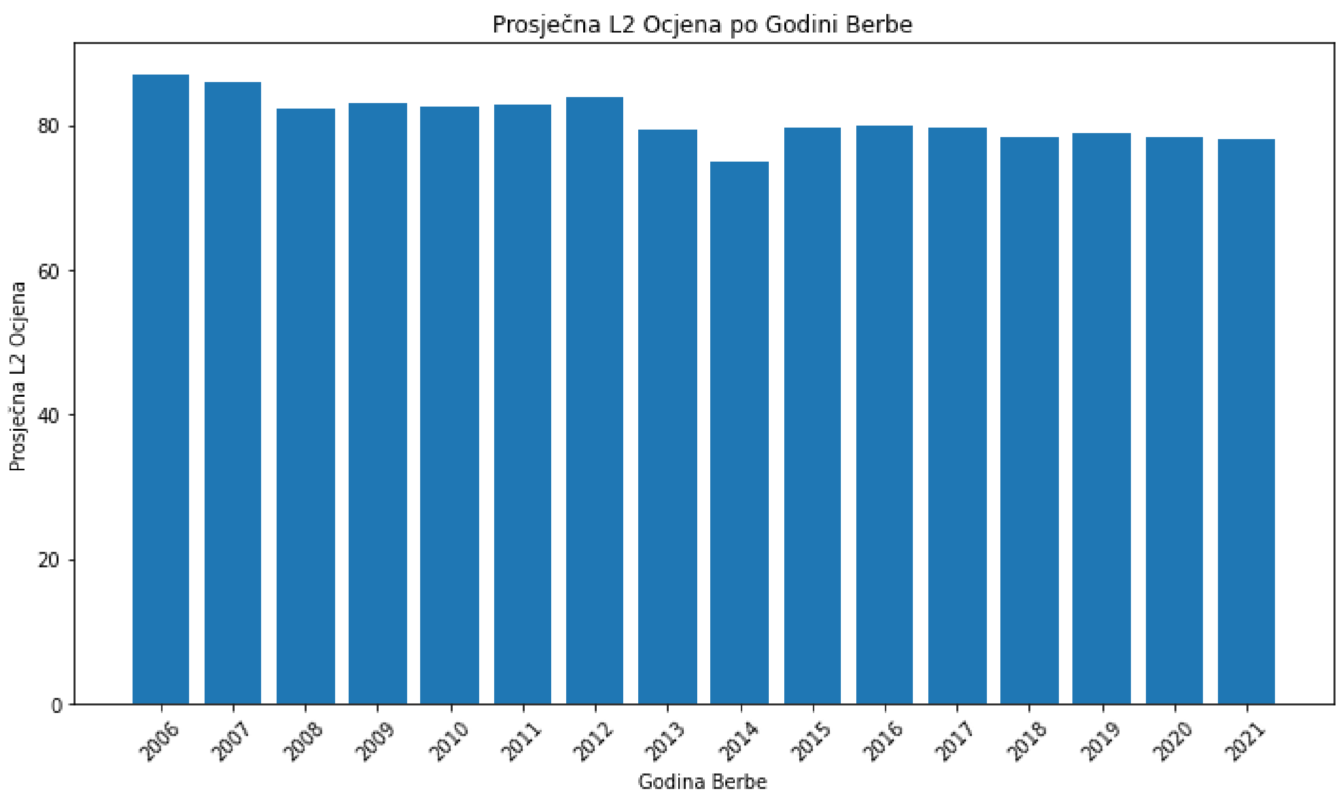


Figure 11 – Average score per year of harvest

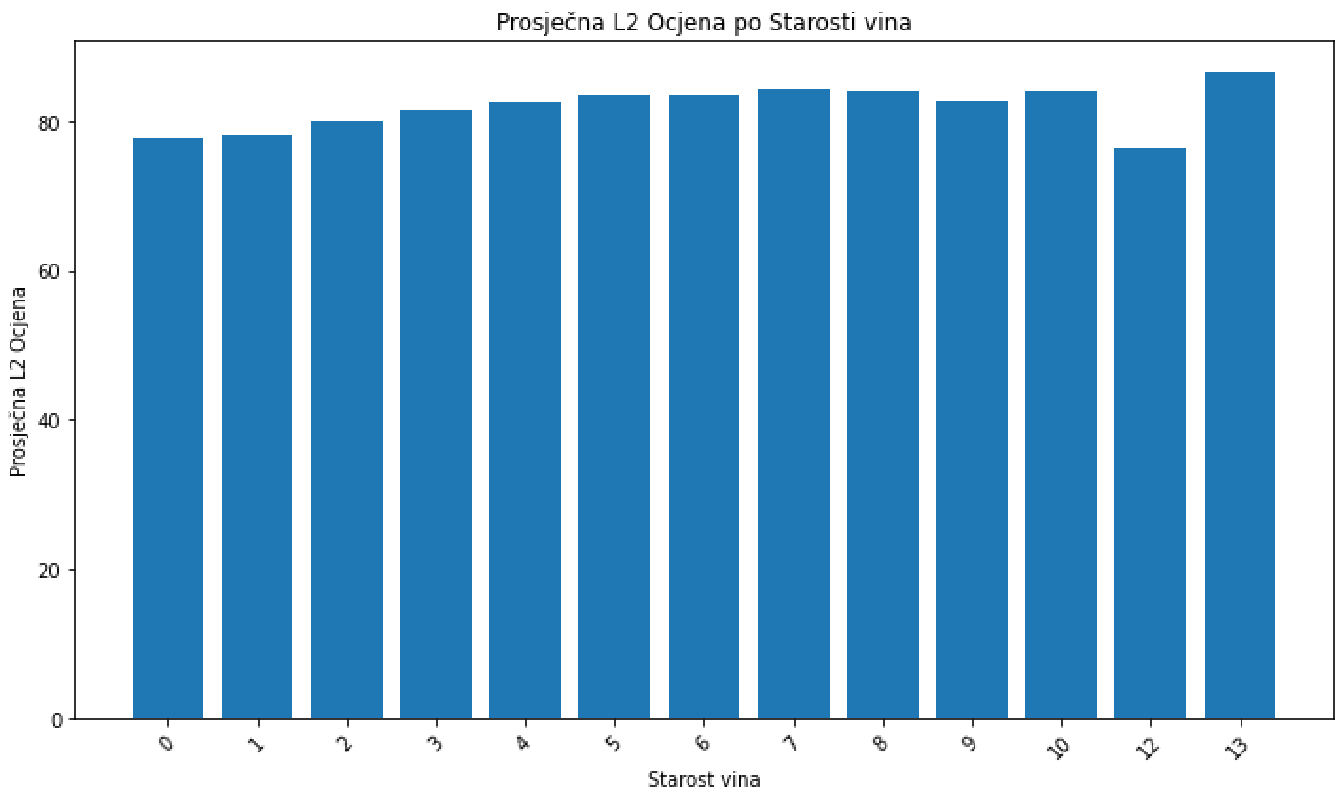


Figure 12 – Average grade by age of wine

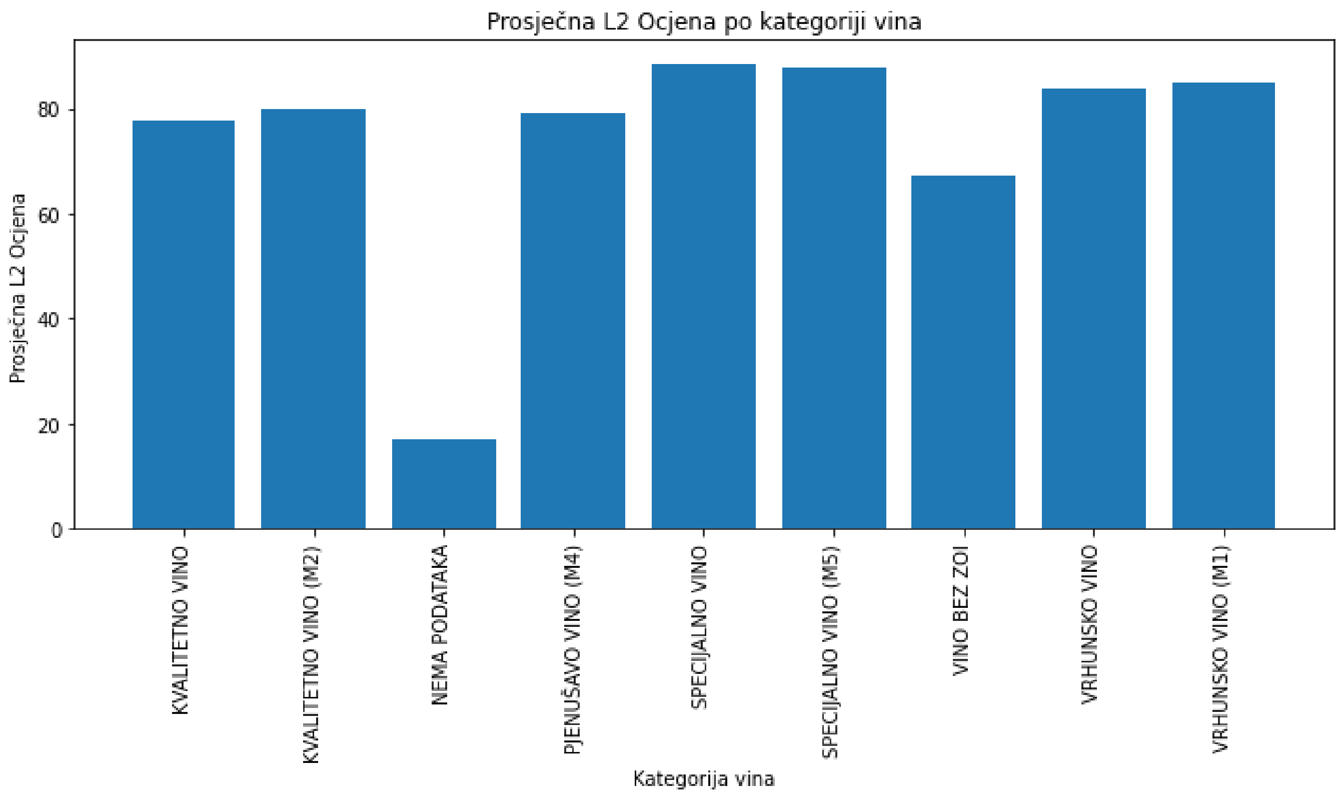


Figure 13 – Average score by wine category

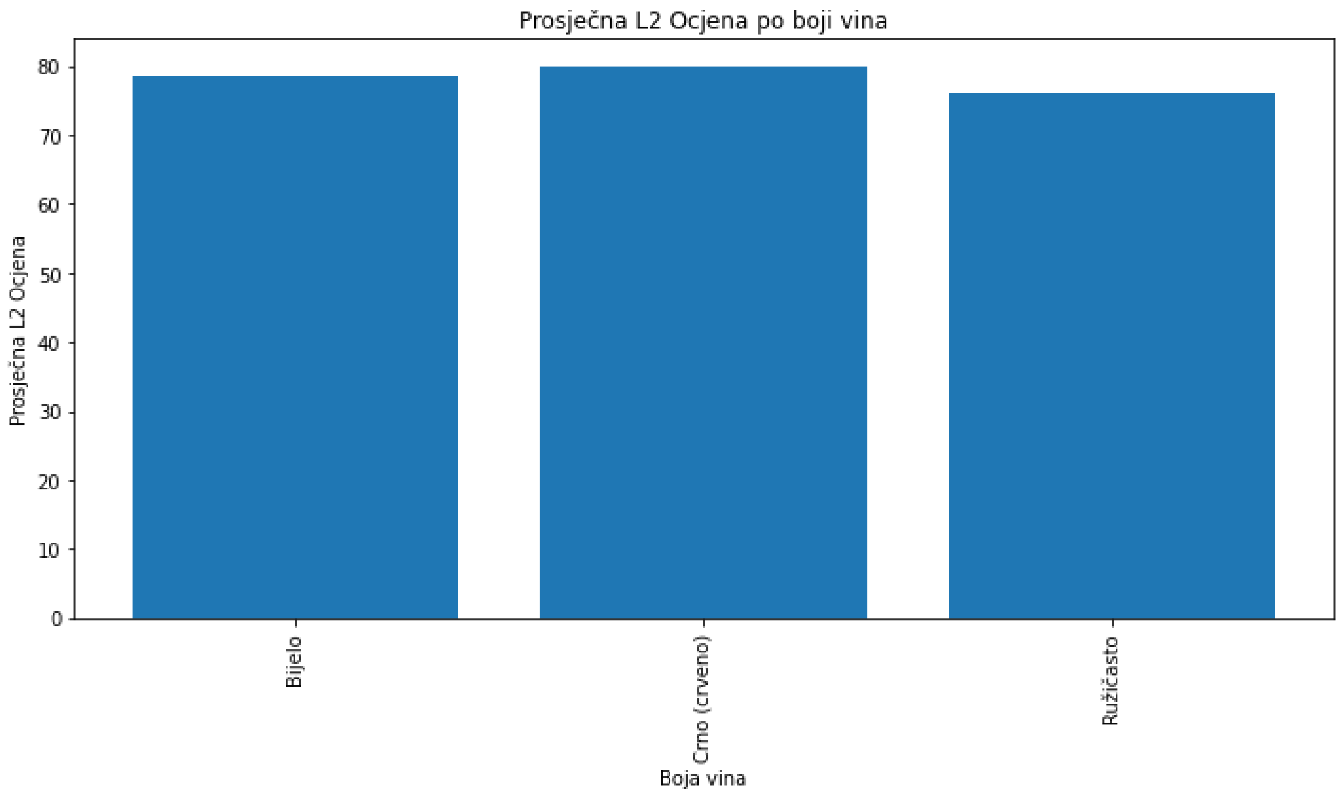
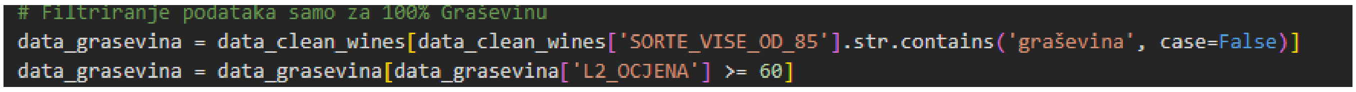


Figure 14 – Average score by wine colour

Training a model for predicting the grade of 100% Graševina

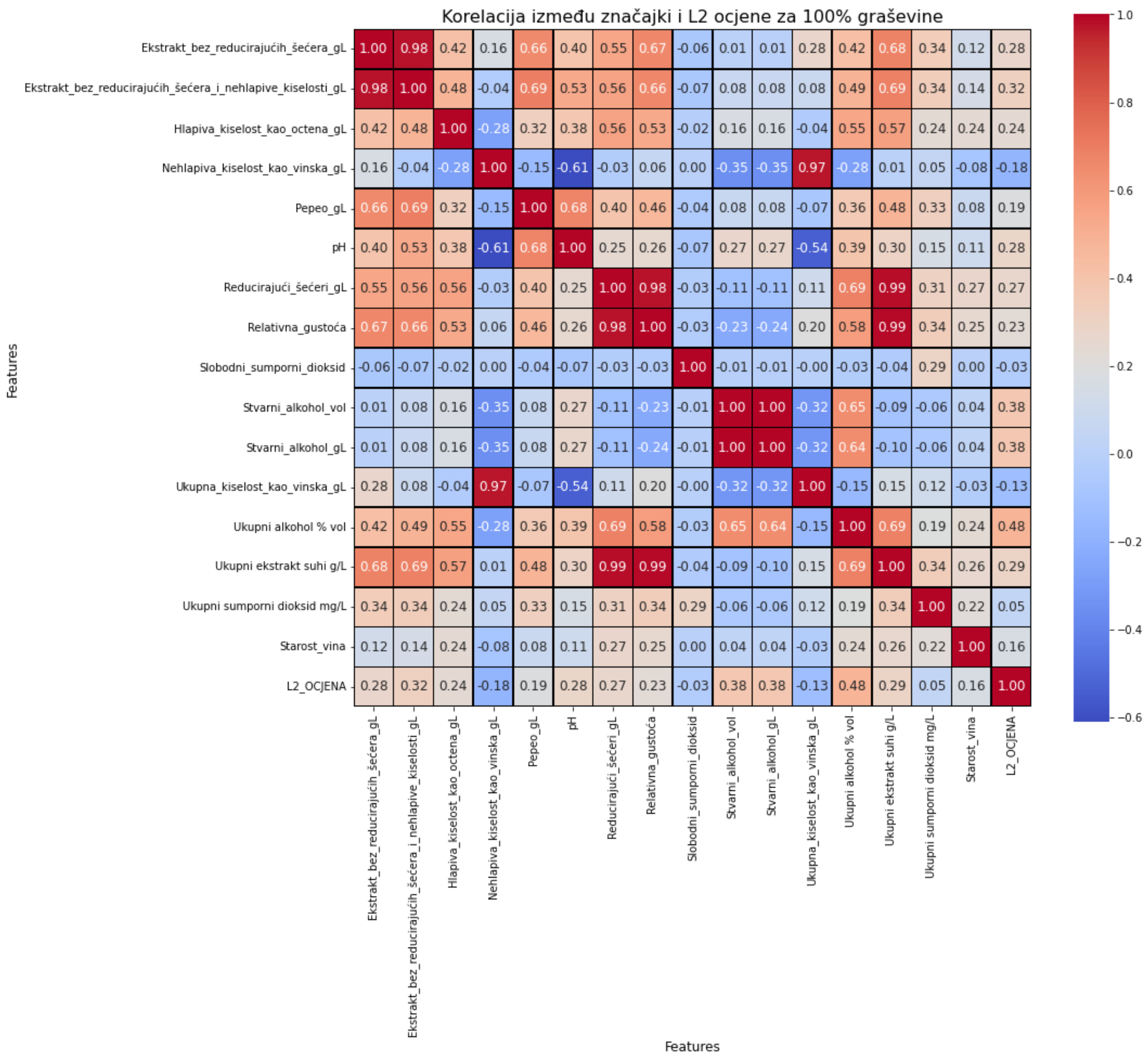
The task is to create a model with the ability to predict the grade of 100% Graševina based on physical and chemical properties. Due to the high amount of ratings in the 70-90 segment, we will train the model to predict a score of 60-100.



```
# Filtriranje podataka samo za 100% Graševinu
data_grasevina = data_clean_wines[data_clean_wines['SORTE_VISE_OD_85'].str.contains('graševina', case=False)]
data_grasevina = data_grasevina[data_grasevina['L2_OCJENA'] >= 60]
```

Figure 15 – Filtering Graševina and grades above 60

In the image below, we can see a correlation matrix for a dataset like this:

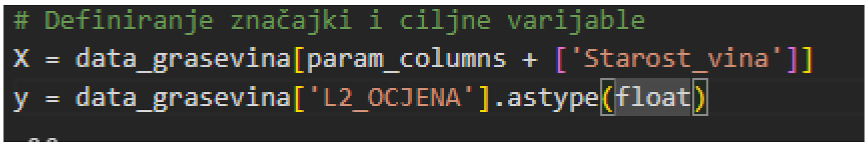


	Ekstrakt_bez_reducirajućih_šećera_gL	Ekstrakt_bez_reducirajućih_šećera_i_nehlapive_kiselosti_gL	Hlapiva_kiselost_kao_octena_gL	Nehlapiva_kiselost_kao_vinska_gL	Pepeo_gL	pH	Reducirajući_šećeri_gL	Relativna_gustoća	Slobodni_sumporni_dioksid	Stvarni_alkohol_vol	Stvarni_alkohol_gL	Ukupna_kiselost_kao_vinska_gL	Ukupni alkohol % vol	Ukupni ekstrakt suhi g/L	Ukupni sumporni dioksid mg/L	Starost_vina	L2_OCJENA
Ekstrakt_bez_reducirajućih_šećera_gL	1.00	0.98	0.42	0.16	0.66	0.40	0.55	0.67	-0.06	0.01	0.01	0.28	0.42	0.68	0.34	0.12	0.28
Ekstrakt_bez_reducirajućih_šećera_i_nehlapive_kiselosti_gL	0.98	1.00	0.48	-0.04	0.69	0.53	0.56	0.66	-0.07	0.08	0.08	0.08	0.49	0.69	0.34	0.14	0.32
Hlapiva_kiselost_kao_octena_gL	0.42	0.48	1.00	-0.28	0.32	0.38	0.56	0.53	-0.02	0.16	0.16	-0.04	0.55	0.57	0.24	0.24	0.24
Nehlapiva_kiselost_kao_vinska_gL	0.16	-0.04	-0.28	1.00	-0.15	-0.61	-0.03	0.06	0.00	-0.35	-0.35	0.97	-0.28	0.01	0.05	-0.08	-0.18
Pepeo_gL	0.66	0.69	0.32	-0.15	1.00	0.68	0.40	0.46	-0.04	0.08	0.08	-0.07	0.36	0.48	0.33	0.08	0.19
pH	0.40	0.53	0.38	-0.61	0.68	1.00	0.25	0.26	-0.07	0.27	0.27	-0.54	0.39	0.30	0.15	0.11	0.28
Reducirajući_šećeri_gL	0.55	0.56	0.56	-0.03	0.40	0.25	1.00	0.98	-0.03	-0.11	-0.11	0.11	0.69	0.99	0.31	0.27	0.27
Relativna_gustoća	0.67	0.66	0.53	0.06	0.46	0.26	0.98	1.00	-0.03	-0.23	-0.24	0.20	0.58	0.99	0.34	0.25	0.23
Slobodni_sumporni_dioksid	-0.06	-0.07	-0.02	0.00	-0.04	-0.07	-0.03	-0.03	1.00	-0.01	-0.01	-0.00	-0.03	-0.04	0.29	0.00	-0.03
Stvarni_alkohol_vol	0.01	0.08	0.16	-0.35	0.08	0.27	-0.11	-0.23	-0.01	1.00	1.00	-0.32	0.65	-0.09	-0.06	0.04	0.38
Stvarni_alkohol_gL	0.01	0.08	0.16	-0.35	0.08	0.27	-0.11	-0.24	-0.01	1.00	1.00	-0.32	0.64	-0.10	-0.06	0.04	0.38
Ukupna_kiselost_kao_vinska_gL	0.28	0.08	-0.04	0.97	-0.07	-0.54	0.11	0.20	-0.00	-0.32	-0.32	1.00	-0.15	0.15	0.12	-0.03	-0.13
Ukupni alkohol % vol	0.42	0.49	0.55	-0.28	0.36	0.39	0.69	0.58	-0.03	0.65	0.64	-0.15	1.00	0.69	0.19	0.24	0.48
Ukupni ekstrakt suhi g/L	0.68	0.69	0.57	0.01	0.48	0.30	0.99	0.99	-0.04	-0.09	-0.10	0.15	0.69	1.00	0.34	0.26	0.29
Ukupni sumporni dioksid mg/L	0.34	0.34	0.24	0.05	0.33	0.15	0.31	0.34	0.29	-0.06	-0.06	0.12	0.19	0.34	1.00	0.22	0.05
Starost_vina	0.12	0.14	0.24	-0.08	0.08	0.11	0.27	0.25	0.00	0.04	0.04	-0.03	0.24	0.26	0.22	1.00	0.16
L2_OCJENA	0.28	0.32	0.24	-0.18	0.19	0.28	0.27	0.23	-0.03	0.38	0.38	-0.13	0.48	0.29	0.05	0.16	1.00

Figure 16 – correlation matrix of all parameters

We see the greatest correlation between the grades with the amount of alcohol in the wine, and the lowest with non-volatile acidity.

A wide range of machine learning algorithms such as linear regression, logistic regression, supporting vector machine (SVM), neural networks, and many more are available for the learning process. Each technique has its own strength and weakness. In this paper, we use the following supervised learning algorithms to predict wine quality
We will define the features for the input and output (targeting) variables for all models. The input variables are all the chemical parameters of the wine, along with the age of the wine, while the target variable is the grade.



```
# Definiranje značajki i ciljne varijable
X = data_grasevina[param_columns + ['Starost_vina']]
y = data_grasevina['L2_OCJENA'].astype(float)
```

Figure 17 – Input and output variants for all models

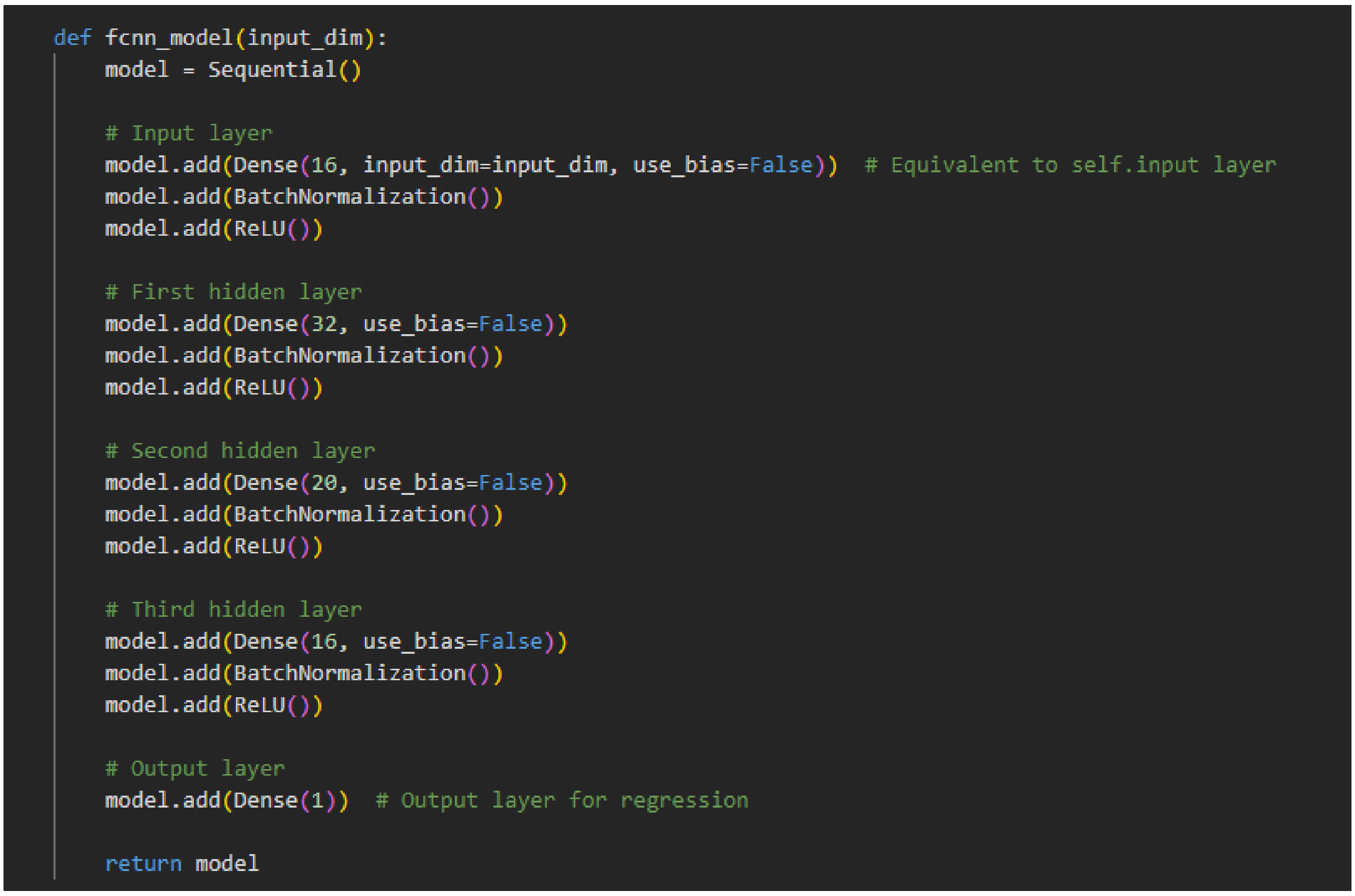
To evaluate the results, we will use MSE (mean squared error), MAE (mean absolute error), and RMSE (root mean squared error), common metrics used to evaluate the performance of regression models. Each metric measures the difference between the predicted values and the actual values in the dataset, but they do so in slightly different ways.

- MSE measures the average of the squared of the difference between the predicted values and the actual values. It punishes major mistakes more severely than smaller ones
- The MAE measures the average of the absolute differences between the predicted values and the actual values. Provides a linear score that doesn't penalize major errors as strongly as MSE
- RMSE: Measures the square root of the average square error; combines the properties of MSE and is in the same units as the target variable
- R2, also known as the coefficient of determination, is the ratio of variance in a dependent variable that is predictable from independent variables. This is a measure of the goodness of fittanj
  The regression model
  - R2= 1: Indicates that the model perfectly predicts the dependent variable. All the variances in the dependent variable explain the independent variables.

- R2= 0: Indicates that the model does not explain any variance in the dependent variable. The model is just as good as simply predicting the mean of the dependent variable.
- R2 < 0: Shows that the model is worse than simply predicting the mean of the dependent variable. This can happen when the model is extremely poor.

1. Neural network

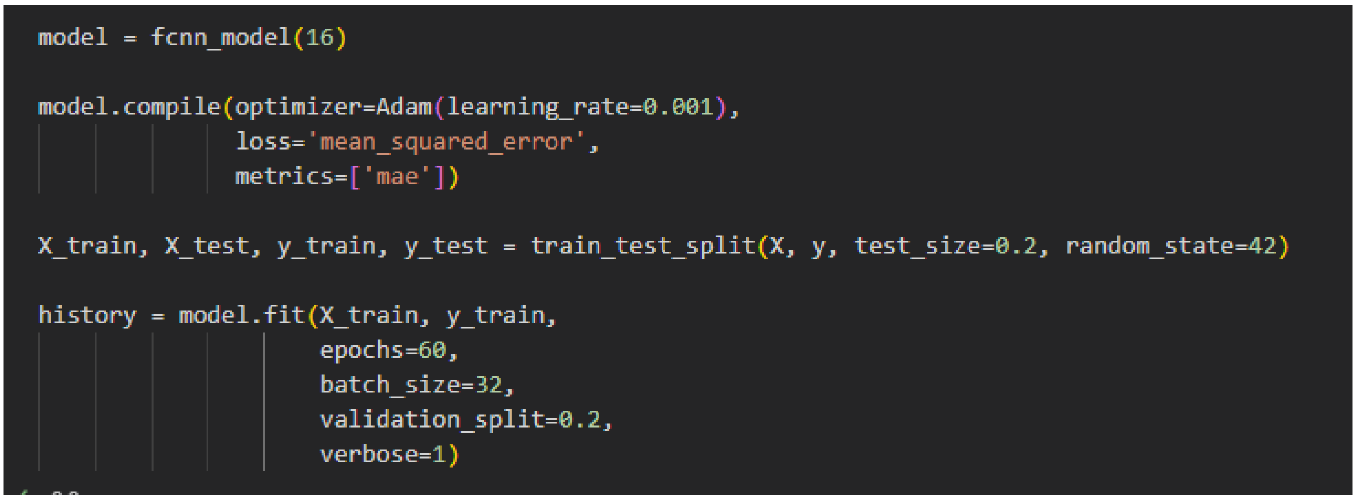
ANNs (Artificial Neural Network) are a very primitive generalization of biological neurons. They are made up of layers of computing units called neurons, with a connection between the different layers through adjustable weights. The main constituents of ANN are weights, bias, and activation function. The excellent choice of activation function results in the correct accuracy of the ANN model. The activation function we use is ReLu. The passage of information through a predetermined path between neurons is the fundamental idea behind the construction of ANN. Its architecture is very flexible, and various network parameters (such as weights, bias, number of nodes, and number of hidden layers) can be adjusted to improve network performance. Neurons can sum up information from multiple sources and apply a non-linear transformation to each node, which helps the network learn the complexity of the data present. By applying linear and nonlinear transformations to the input data, ANNs transform these initial representations to a specific outcome. Depending on the learning task, the outcome of the network can be either classification or regression. The neural network we use is defined as follows:



```python
def fcnn_model(input_dim):
    model = Sequential()

    # Input layer
    model.add(Dense(16, input_dim=input_dim, use_bias=False))  # Equivalent to self.input layer
    model.add(BatchNormalization())
    model.add(ReLU())

    # First hidden layer
    model.add(Dense(32, use_bias=False))
    model.add(BatchNormalization())
    model.add(ReLU())

    # Second hidden layer
    model.add(Dense(20, use_bias=False))
    model.add(BatchNormalization())
    model.add(ReLU())

    # Third hidden layer
    model.add(Dense(16, use_bias=False))
    model.add(BatchNormalization())
    model.add(ReLU())

    # Output layer
    model.add(Dense(1))  # Output layer for regression

    return model
```

Figure 18 – Definition of a neural network

By using the Dense layer, we fulfill the full connectivity of the in which case we call the ANN FCNN (Fully Connected Neural Network), which means that all the nodes from the adjacent layers are connected to each other with the exception of their own loops and nodes from the same layer. It has an input layer (16 neurons for 16 input parameters), three hidden layers (32, 20, 16 neurons), and an output layer (1 neuron – rating).



```
model = fcnn_model(16)

model.compile(optimizer=Adam(learning_rate=0.001),
              loss='mean_squared_error',
              metrics=['mae'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

history = model.fit(X_train, y_train,
                    epochs=60,
                    batch_size=32,
                    validation_split=0.2,
                    verbose=1)
```

Figure 19 – Neural network training

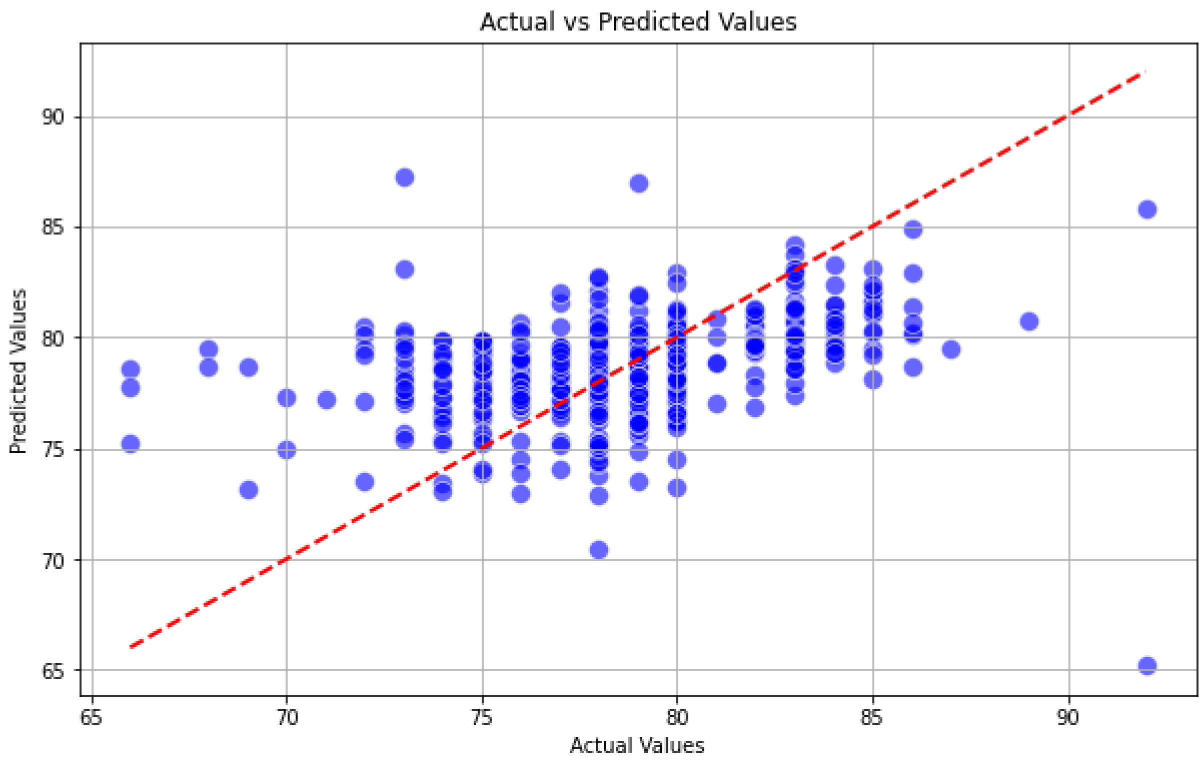


Figure 20 – Comparison of actual rating and prediction

R-squared: 0.051749983152156 29

MSE: 11.9868

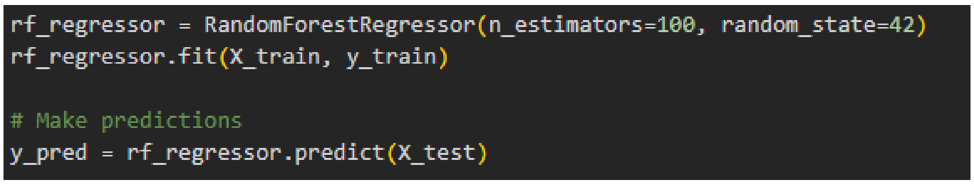
IEA: 2.5442

RMSE: 3.4622

Accuracy 100*(1-RMSE/40): 91.3445%

2. Random Forest algorithm

The Random Forest algorithm is a learning technique that constructs a multitude of decision trees during training and aggregates their results to improve accuracy and control over-equipping. Each decision tree is trained on random subsets of data with a random subset of features considered for each division, which introduces diversity among the trees. For classification tasks, the final prediction is determined by majority voting in all trees, while for regression tasks, it is the average of all tree predictions. This approach improves the robustness of the model and reduces variance, making Random Forest efficient for handling complex datasets with noise and nonlinearity. Its ability to provide feature relevance also helps to understand which features have the greatest impact on predictions



```
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

# Make predictions
y_pred = rf_regressor.predict(X_test)
```

Figure 21 – Random Forest Regressor

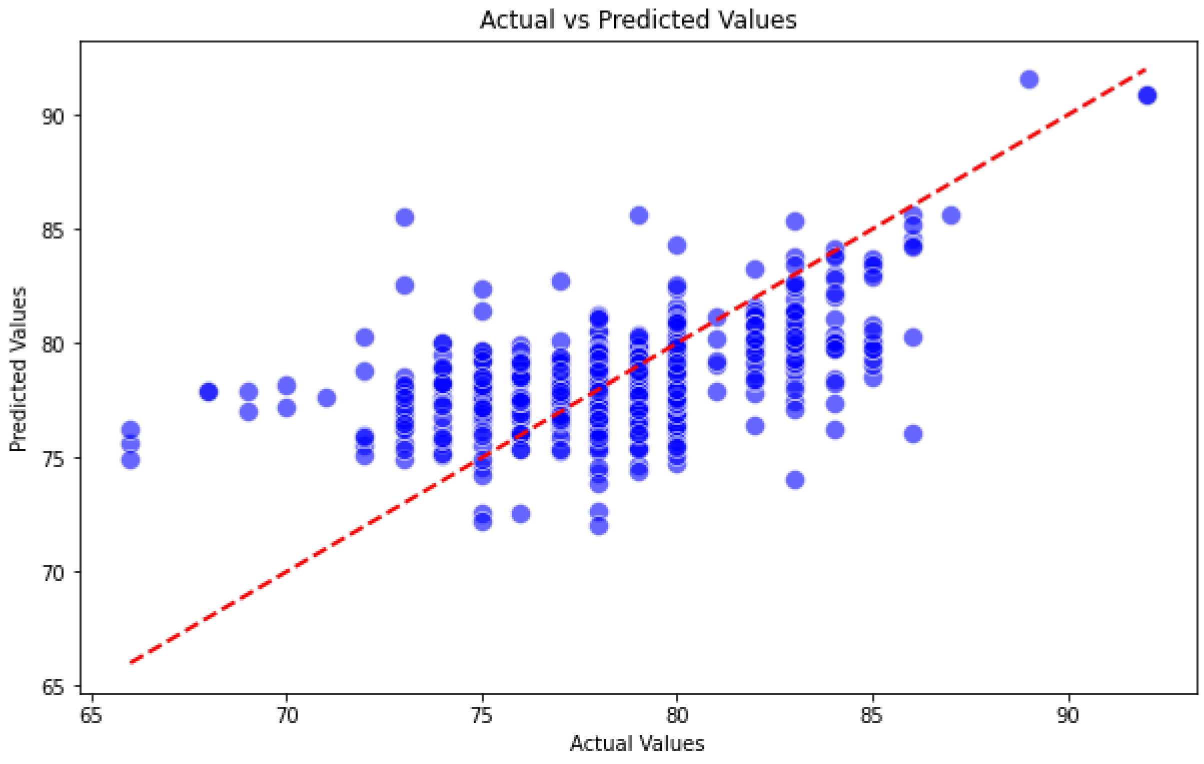


Figure 22 – Comparison of actual rating and prediction

R-squared: 0.2766389805668227

MSE: 9.1440

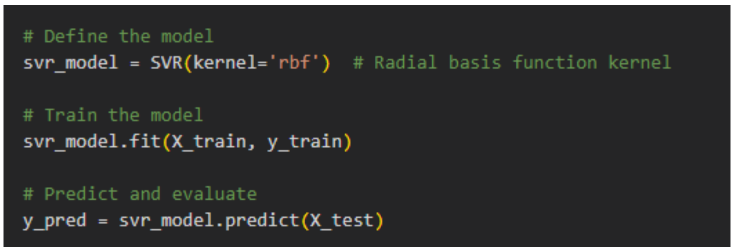
IEA: 2.2989

RMSE: 3.0239

Accuracy 100*(1-RMSE/40): 92.4402%

3. SVR (Support Vector Regression)

Support Vector Regression (SVR) is a type of Support Vector Machine (SVM) used for regression tasks. It aims to find a function that matches data within a certain margin of tolerance, known as an epsilon while balancing model complexity and errors. The SVR model minimizes the combination of prediction error and the control member that controls the complexity of the model. It uses support vectors—data points that lie beyond the epsilon margin—to define the regression function, making the model robust to deviations and noise. SVR can handle nonlinear relationships using kernel functions, such as the radial base function (RBF), which maps data to multidimensional spaces where linear regression can be applied. This makes SVR a flexible and powerful tool for various regression problems.



```
# Define the model
svr_model = SVR(kernel='rbf')  # Radial basis function kernel

# Train the model
svr_model.fit(X_train, y_train)

# Predict and evaluate
y_pred = svr_model.predict(X_test)
```

Figure 23 – SVR

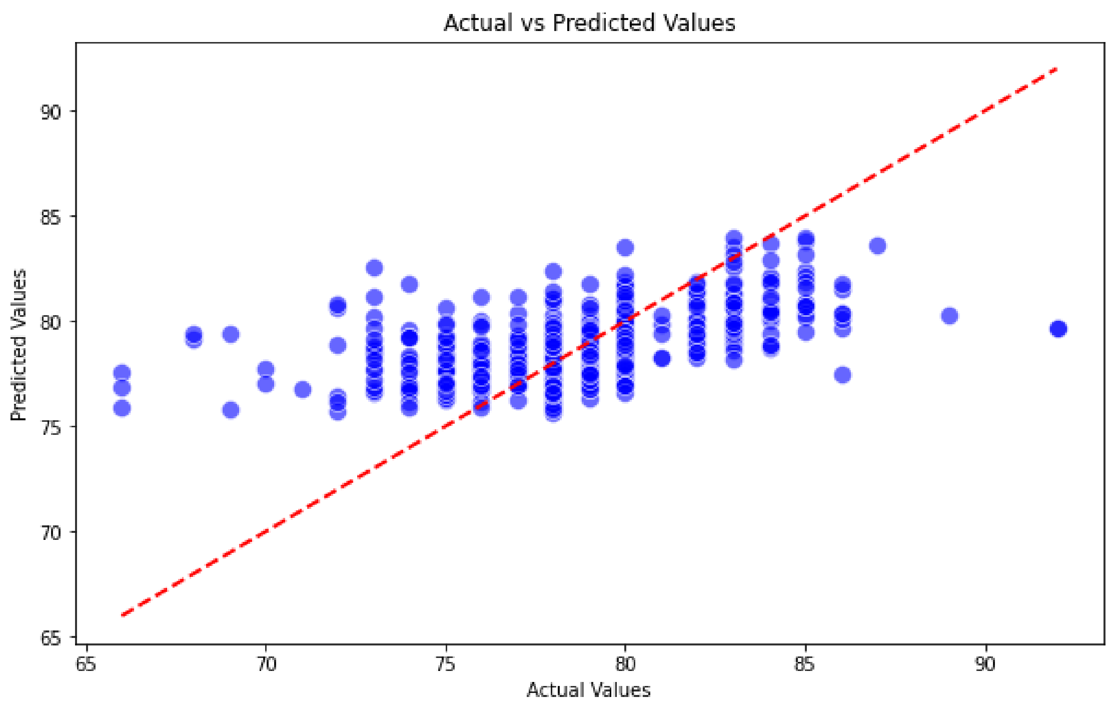


Figure 24 – Comparison of actual rating and prediction

R-squared: 0.26250342333850885

MSE: 9.3227

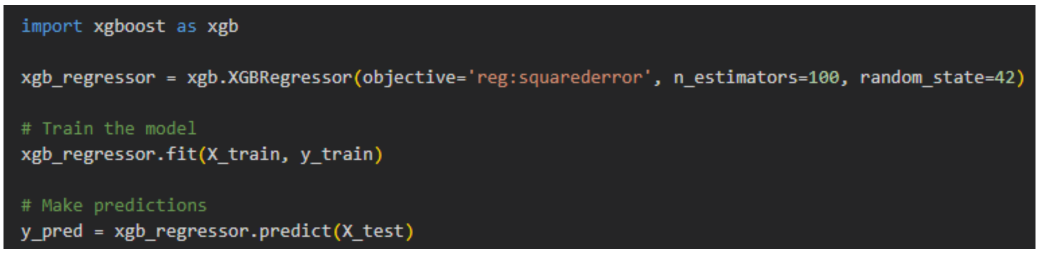
IEA: 2.2581

RMSE: 3.0533

Accuracy 100*(1-RMSE/40): 92.3667%

4. XGBoost (Extreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting) is an advanced implementation of gradient boosting that excels in classification and regression tasks. It builds a series of decision trees, where each tree corrects the mistakes of its predecessors, resulting in a powerful model. XGBoost optimizes performance through techniques such as regularization, which prevents over-provisioning, and robust handling of missing values. It uses a framework to increase the gradient, where the model is iteratively improved by minimizing the loss function by lowering the gradient. With its ability to efficiently handle large data sets and support for various hyperparameters and scaling strategies, XGBoost is known for its high prediction accuracy and scalability in machine learning competitions and real-world applications



```
import xgboost as xgb

xgb_regressor = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, random_state=42)

# Train the model
xgb_regressor.fit(X_train, y_train)

# Make predictions
y_pred = xgb_regressor.predict(X_test)
```

Figure 25 – XGBoost Regressor

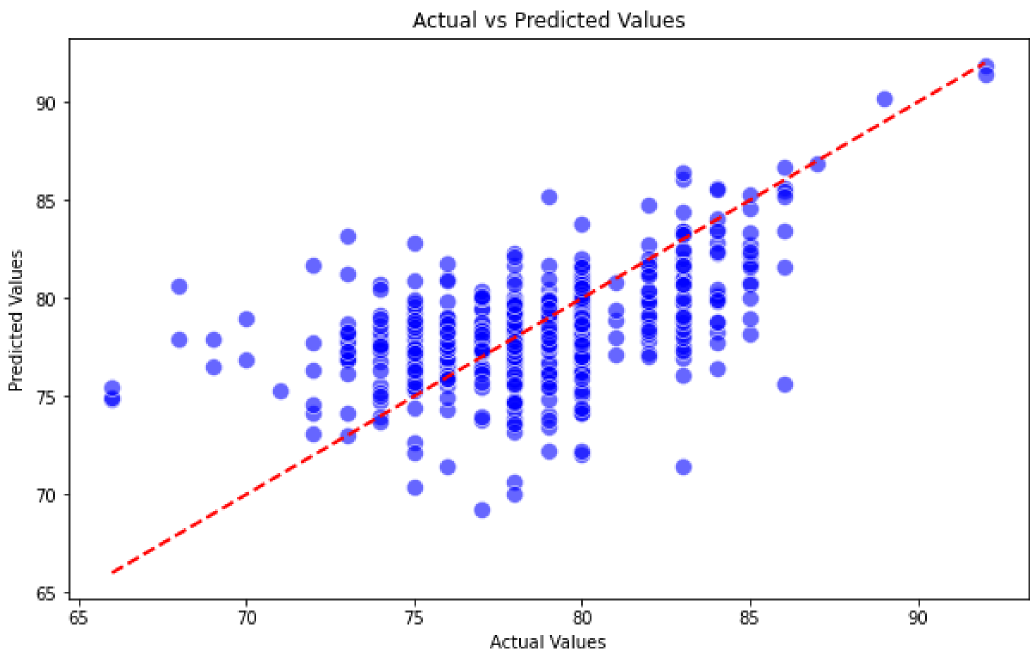


Figure 26 – Comparison of actual rating and prediction

R-squared: 0.18407779118779033

MSE: 10.3140

IEA: 2.4118

RMSE: 3.2115

Accuracy 100*(1-RMSE/40): 91.9711%

Results

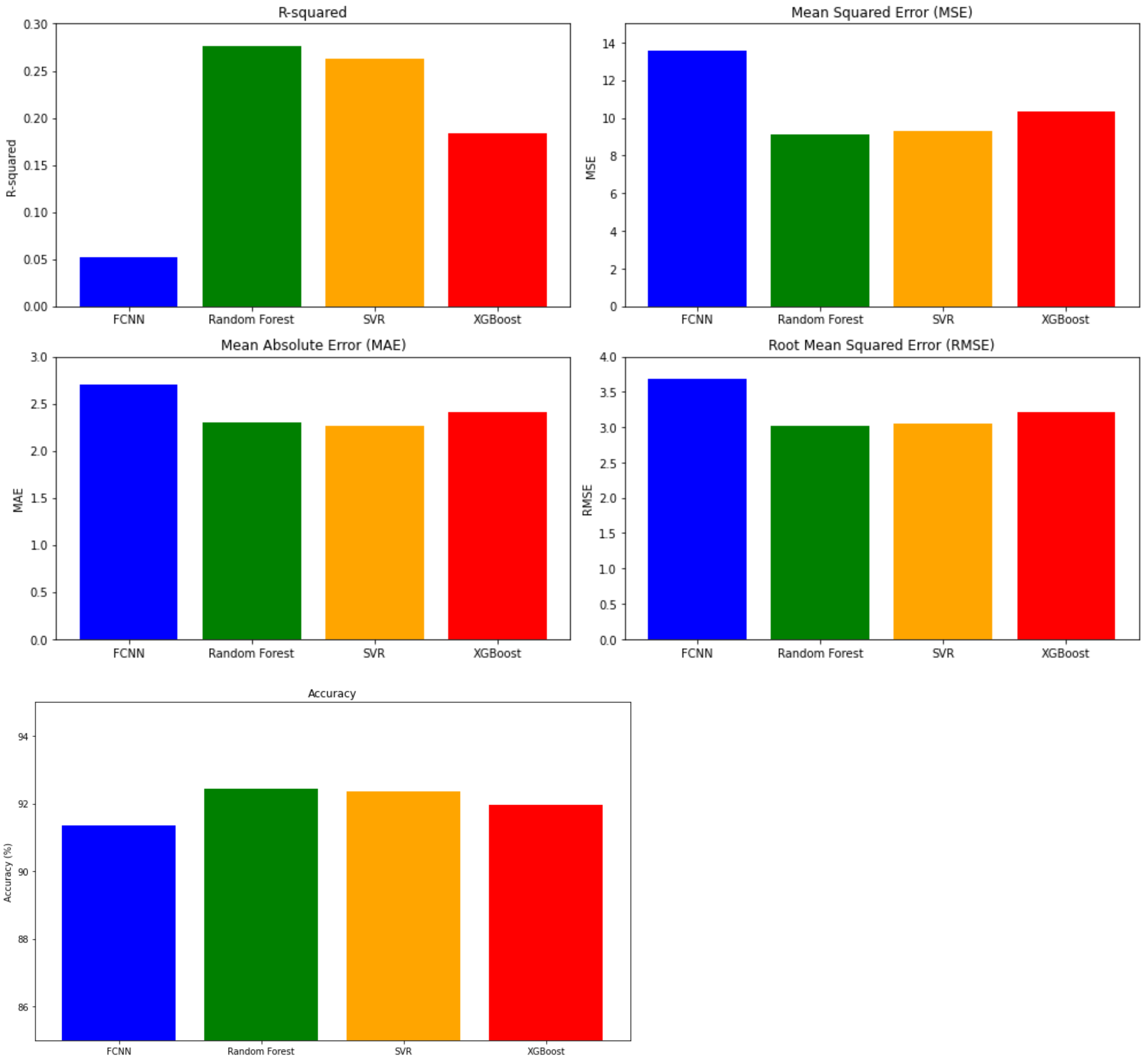


Figure 27 – Graphs with results

Random Forest has the largest R2, indicating that it explains the greatest variance. FCNN is the worst in this regard. Random Forest has the lowest MSE, indicating that it performs best in terms of minimizing the squares of the differences between predictions and actual values. FCNN has the highest MSE. The SVR has the lowest MAE, which means it has the lowest average prediction error. FCNN has the highest MAE, indicating higher average errors. Random Forest has the lowest RMSE, showing the smallest error size. FCNN has the highest RMSE. Random Forest has the highest accuracy value, followed closely by SVR. All models show high accuracy, but Random Forest has a slight advantage. Random Forest has proven to be the best choice overall, given its combination of high accuracy, low error metric (MSE and RMSE), and reasonable R-squared value.

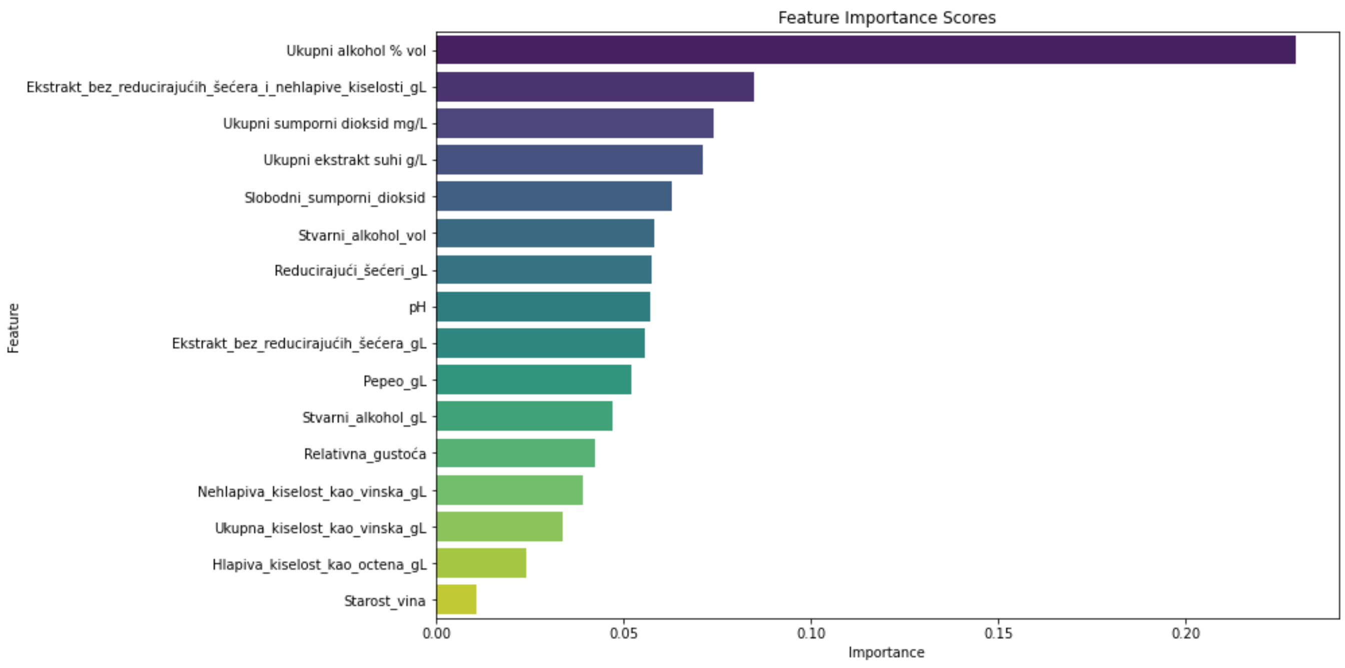


Figure 28 – Feature Significance Graph

When we draw the significance of all the features for the Random Forest model, we see that the most important feature for wine quality control is alcohol. Which makes perfect sense because it's not just about the feelings after drinking, but it affects the taste, texture and structure of the wine itself. Interestingly, the age of the wine contributes the least to a good score among other parameters, which makes sense because in the dataset all ages of wine (0-13 years) have roughly the same average score.

Conclusion

This paper shows that different statistical analyses can be used to analyze the parameters in an existing dataset to determine the quality of wine. On the basis of various analyses, the quality of wine can be predicted before its production.

According to the test results, Random Forest proved to be the best model for predicting wine quality. It has the largest R2, which indicates that it explains the greatest variance in the data. Also, Random Forest has the lowest MSE, which suggests the best performance in terms of minimizing the squares of the differences between predictions and actual values. In contrast, FCNN shows the worst results in this regard with the highest MSE.

When other metric scores are compared, the SVR stands out as the model with the lowest MAE, which means it has the lowest average prediction error. On the other hand, FCNN shows the highest MAE, indicating higher average errors. Random Forest has the lowest RMSE, which shows the smallest error size, whereas FCNN has the highest RMSE.

In terms of accuracy, Random Forest has the highest accuracy value, with the SVR following right behind it. All models show high accuracy, but Random Forest has a slight advantage in this metric comparison. Based on a combination of high accuracy, low error metrics (MSE and RMSE), and a reasonable R-squared value, Random Forest has proven to be the best choice overall.

When we draw the significance of all the features for the Random Forest model, we see that the most important feature for wine quality control is alcohol. This is logical, given that alcohol affects the taste, texture and structure of wine. On the other hand, the age of the wine contributes the least to a good score among other parameters, which can be explained by the fact that in the dataset all wine ages (0-13 years) have roughly the same average score.

This paper shows an alternative approach that could be used to obtain the quality of wine and can therefore be a good starting point for reviewing the variables on which the quality of wine *depends.*

Literature

[1] https://spiralizing.github.io/DSEntries/WineQuality/

[2] https://medium.com/@m.ariefrachmaann/wine-quality-prediction-with-machine-learning-model-10c29c7e3360

[3] https://www.scirp.org/journal/paperinformation?paperid=107796#t3

[4] Mor, N. S., Asras, T., Gal, E., Demasia, T., Tarab, E., Ezekiel, N., ... Mor, O. (2022, February 1). Wine Quality and Type Prediction from Physicochemical Properties Using Neural Networks for Machine Learning: A Free Software for Winemakers and Customers. https://doi.org/10.31222/osf.io/ph4cu