Introduction to Machine Learning
Assignment 1

Andrei Medesan (S5130727) &
Janan Jahed (S5107318) &
Natalia Gayoso (S5221218)
Group 12

September 24, 2023

Part I - Feature Types

1. A random number in range [0, 1] → **Continuous**, as it can take any real value in the given range [0,1].
2. Number of bikes in Groningen → **Discrete**, as the numbers can be ordered and are countable
3. Handedness → **Categorical**, as no numerical order can be established
4. Current types of Machine Learning methods → **Categorical**
5. Duration of the trip from home to campus Zernike → **Continuous**

Part II - Classifiers vs Regressors

1. Regarding the prediction of the success rate of students in a course, there are several features that must be considered in order to assess a positive outcome for peers. Thus, the most useful features to take into account are *number of study hours*, *number of sleeping hours*, and *number of times student attended lectures*. These features would yield the most useful results for the student in order to pass a course.

   Regarding the *number of study hours*, it is a highly important feature since more hours of study are correlated to better performance for the student in their final assessment. Therefore, this feature is likely to predict the success rate.

   Regarding the *number of sleeping hours* indicates another relevant feature to consider since sleeping aids people in memory consolidation. It also has an impact on the overall activity of the student as the lack of sleep can cause a negative effect on the student's performance. Thus, this feature represents a predictor for the success rate.

   Ultimately, the *number of times student attended lectures* represents a key feature for the student's performance. This is due to the fact that attending lectures can help students gain valuable information that might aid them in their final assessment.

2. Predicting the success rate of students for passing a course can be either a classification or regression task depending on the purpose of defining success. This is because classification refers to a double manner for categorizing items with respect to the prediction that must

be assessed, whilst regression is a continuous variable that requires numerical values (Otten, 2023).

Consequently, it is necessary to determine whether the success rate should be assessed in terms or pass or fail, which would be a classification task, or it would be an average grade from 1 to 10, which would be a regression task.

Part III - Feature Transformations and Linear Separability

a. Before starting the training process we were instructed to choose the features that best describe the quality of a wine. To decide upon these features, we constructed a heat map (Figure 1) in Python to visualize the pairwise correlations between the features in the wine data set. The correlation coefficient between two features is shown by each cell in the heat map. The features that lay in the extremes (positive or negative values) with respect to *target* are the features that are most important in determining wine quality. Hence, after looking at the heat map and the correlation coefficient, the following 6 features were chosen:

1. *flavonoids*
2. *od280/od315_of_diluted_wines*
3. *total_phenols*
4. *proline*
5. *hue*
6. *alcalinity_of_ash*

```
    #The code below shows all the libraries we used for this assignment
    from sklearn import datasets
    import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import classification_report
    from sklearn.preprocessing import StandardScaler
    from sklearn.datasets import load_wine
```

```

wine = datasets.load_wine()
wine_df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
wine_df['target'] = wine.target

X = wine_df.drop('target', axis=1)
y = wine_df['target']

# Calculate the correlation matrix
correlation_matrix = wine_df.corr()

# Plot the heatmap of the correlation matrix
plt.figure(figsize=(15, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
print(feature_names)
```

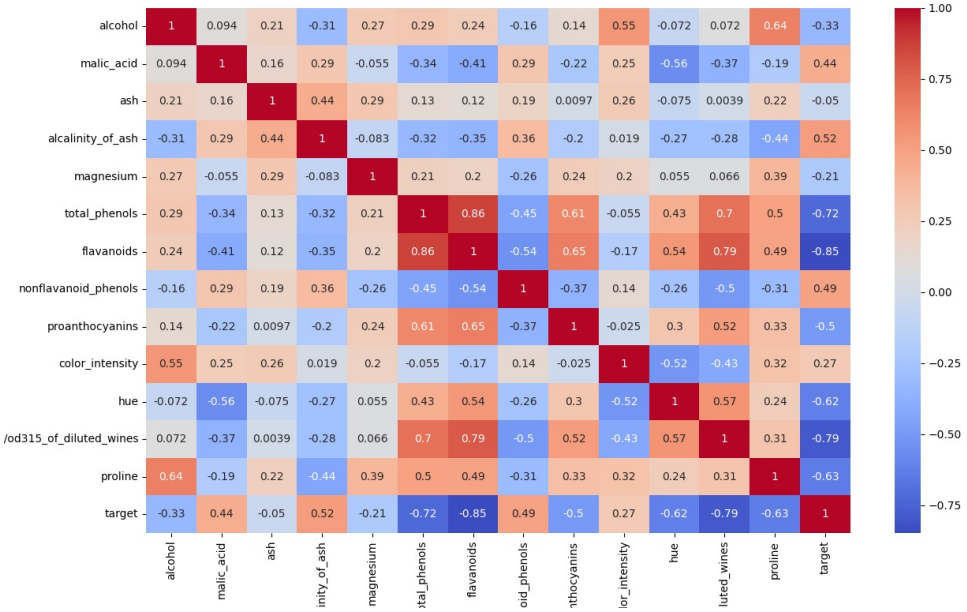


Figure 1: This figure shows a heat map that provides the pairwise correlations between the data set's variables (University of Groningen, n.d.-a).

Moreover, to ensure that these observations are correct, we implemented a code that sorts the features based on their correlation coefficient in descending order. The reason we selected *7* in *head(7)* is because the first printed feature would be the target which we do not need. The results show that our observations were correct.

```
# Getting the absolute correlation coefficients between features vs. the target
    variable, as shown in our heatmap
correlation_with_target = wine_df.corr()['target'].abs()
# Sort the features from high to low - https://pandas.pydata.org/docs/reference/
    api/pandas.DataFrame.sort_values.html
best_features = correlation_with_target.sort_values(ascending=False).head(7)
# Print it in the terminal to see
print(best_features)
```

```
target                          1.000000
flavanoids                      0.847498
od280/od315_of_diluted_wines    0.788230
total_phenols                   0.719163
proline                         0.633717
hue                             0.617369
alcalinity_of_ash               0.517859
```

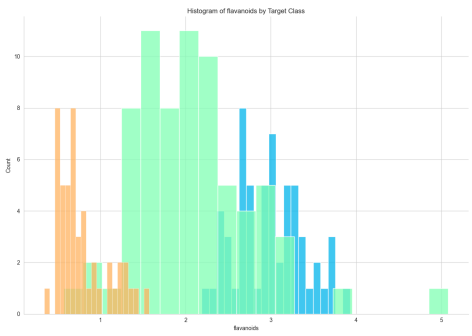
Figure 2: This figure shows the results obtained from implementing the code.

b. **All features** - The code below represents the training process for the 6 features (University of Groningen, n.d.-b).

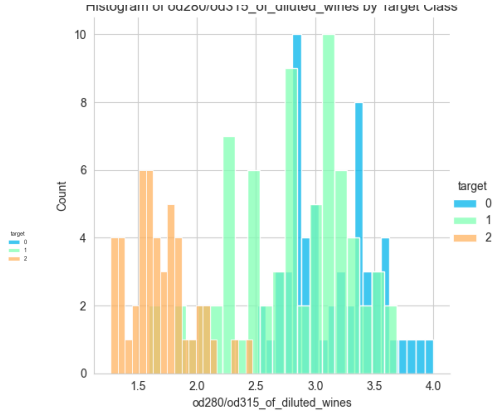
To work with the wine data set, this code employs the libraries *scikit-learn* and *Pandas*. It loads the data set first, then organizes it into a Data Frame, and uses *StandardScaler* to normalize the 6 features (as was advised in the lecture slides). We started this process by creating an array that holds the 6 most important features that we obtained in part A and proceeded to normalize the data. After this step, the normalized 6 features are then divided into training and testing sets, with 70% set aside for training (As was advised in the lecture). On the training data, a logistic regression model is created and trained. To evaluate the model's performance, predictions are produced based on the testing data, and a classification report is printed so we can observe and evaluate our trained model. After running our model, we obtained an accuracy of 98% and decided to choose the given values as the final random_state (42) and test_size (0.3). Lastly, to analyze whether these features are linearly separable, we plotted the histogram of the data to visually observe how the data looks like.

```
1
2  wine = datasets.load_wine()
3  wine_df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
4  wine_df['target'] = wine.target
5
6  X = wine_df.drop('target', axis=1)
7  y = wine_df['target']
8
9  # Select the features you want to use
10 selected_features = ['flavanoids', 'od280/od315_of_diluted_wines', '
       total_phenols', 'proline', 'hue', 'alcalinity_of_ash']
11
12 # Normalize the features
13 scaler = StandardScaler()
14 X_normalized = scaler.fit_transform(wine_df[selected_features])
15
16 X_train, X_test, y_train, y_test = train_test_split(X_normalized, y, test_size
       =0.3, random_state=42)
17
18 logistic_model = LogisticRegression()
19 logistic_model.fit(X_train, y_train)
20 predictions = logistic_model.predict(X_test)
21
22 print(classification_report(y_test, predictions))
23
24
25 # Set the style
26 sns.set_style('whitegrid')
27
28 # Loop through each selected feature to create histograms
29 for feature in selected_features:
30     g = sns.FacetGrid(wine_df, hue="target", palette="rainbow", height=5)
31     g.map(sns.histplot, feature, bins=20, kde=False)
32     g.set(title=f'Histogram of {feature} by Target Class')
33     g.add_legend()
34     plt.show()
```

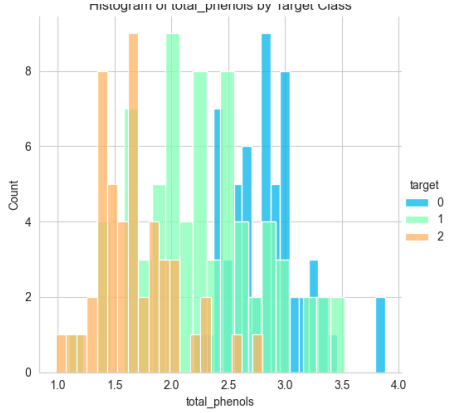
After plotting the 6 histograms (Figure 3) we observed the features closest to linear separability are *flavanoids* and *proline*.



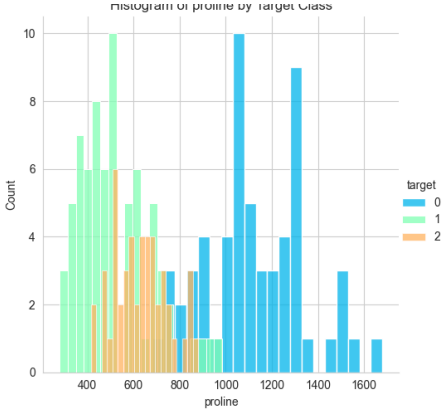
(a) Histogram of the *flavanoids* feature



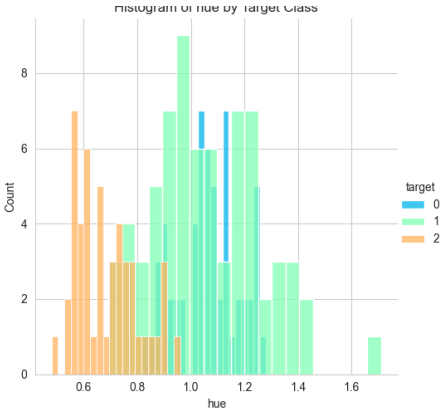
(b) Histogram of the *diluted_wines* feature



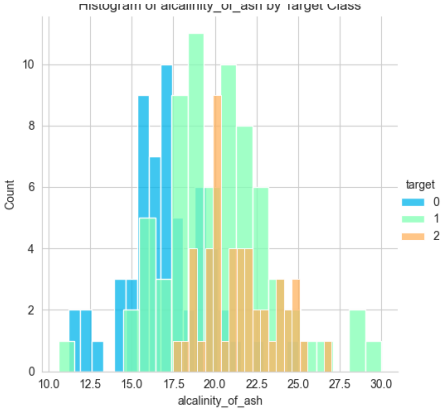
(c) Histogram of the *total_phenols* feature



(d) Histogram of the *proline* feature



(e) Histogram of the *hue* feature



(f) Histogram of the *alcalinity_of_ash* feature

Figure 3: All the six figures represent the histograms generated by the code with the 6 selected features.

```
wine = datasets.load_wine()
wine_df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
wine_df['target'] = wine.target

X = wine_df.drop('target', axis=1)
y = wine_df['target']

# Normalize the features
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_normalized, y, test_size
    =0.3, random_state=42)

logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
predictions = logistic_model.predict(X_test)

print(classification_report(y_test, predictions))
```

Regarding the logistic regression training model, The accuracy for both was 98%, although the precision changed, for example, for class 1 the precision was 95% with 6 features, but it increased to 100% with all features. This could be because there is a greater number of linearly separable features in the complete set, additionally to the higher number of features making a more descriptive set, which results in a higher level of precision.

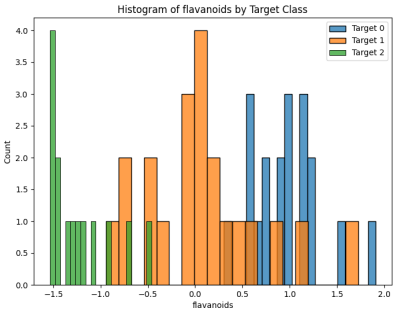
c. The code below represents the model that predicts the wine quality through the use of the six features selected to which simple transformations were implemented. Following the selection of the features, and the normalization of their values, the variable *transformed_features* was applied to the code in order for the model to use transformed features. In this way, it was possible to apply different mathematical functions (Yenigün, 2022) to each of them and regard whether the graphs will exhibit linear separability.

Further on, the model was trained using the transformed features and the logistic regression technique was applied. The results of the model were given through the classification report which stated that the model predicts the wine quality with a 96% accuracy. This shows that the model had some changes and it raised a challenge for the program to predict the wine quality. However, the difference is relatively small which may not be a significant concern since the model can still accurately make predictions.

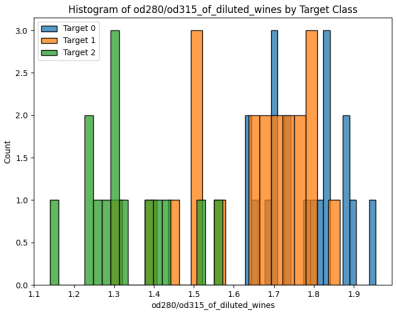
```
wine = datasets.load_wine()
wine_df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
wine_df['target'] = wine.target

X = wine_df.drop('target', axis=1)
y = wine_df['target']

# Select the features you want to use
selected_features = ['flavanoids', 'od280/od315_of_diluted_wines', '
    total_phenols', 'proline', 'hue', 'alcalinity_of_ash']

transformed_features = pd.DataFrame()

# multiplying by 2 transformation
transformed_features['flavanoids'] = 2 * wine_df['flavanoids']

# square root transformation
```

```
transformed_features['od280/od315_of_diluted_wines'] = np.sqrt(wine_df['od280/
    od315_of_diluted_wines'])

# logarithmic transformation
transformed_features['total_phenols'] = np.log1p(wine_df['total_phenols'])

# inverse transformation
transformed_features['proline'] = 1 / wine_df['proline']

# exponential transformation
transformed_features['hue'] = np.exp(wine_df['hue'])

# power of 2 transformation
transformed_features['alcalinity_of_ash'] = wine_df['alcalinity_of_ash'] ** 2

scaler = StandardScaler()
X = scaler.fit_transform(transformed_features)

y = wine.target

# train the model with the transformed features
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=42)

logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)

# give the accuracy of the model with the transformations applied
predictions = logistic_model.predict(X_test)
print(classification_report(y_test, predictions))

# plot the histograms using transformed features
for transformed_feature in transformed_features.columns:
    plt.figure(figsize=(8, 6))
    for target_class in wine_df['target'].unique():
        indices = np.where(y_test == target_class)
        sns.histplot(X_test[indices][:, transformed_features.columns.get_loc(
    transformed_feature)], bins=20, kde=False, label=f'Target {target_class}')
    plt.title(f'Histogram of {transformed_feature} by Target Class')
    plt.xlabel(transformed_feature)
    plt.legend()
    plt.show()
```

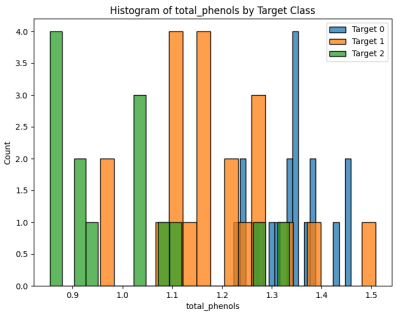
The histograms were implemented using the documentation provided and information from *Real Python* (Solomon, 2018). In this way, the histograms plotted by the code can be observed in Figure 4. Moreover, it can be regarded the linear separability in each of the features transformed, especially in the histograms *flovanoids* and *proline*.



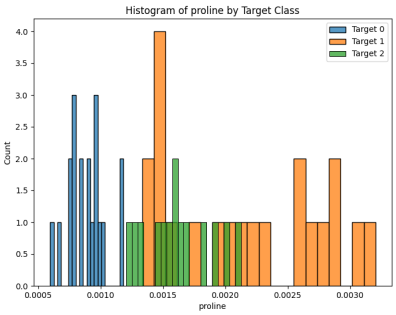
(a) Histogram of the *flavanoids* feature after applying the transformation.



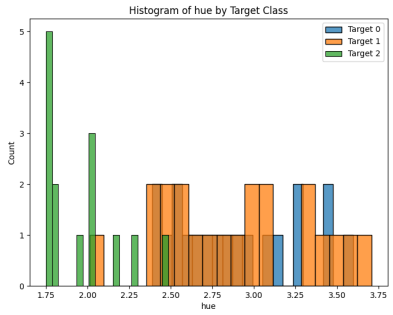
(b) Histogram of the *diluted_wine* feature after applying the transformation.



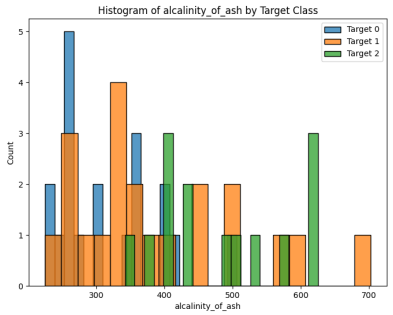
(c) Histogram of the *total_phenols* feature after applying the transformation.



(d) Histogram of the *proline* feature after applying the transformation.



(e) Histogram of the *hue* feature after applying the transformation.



(f) Histogram of the *alcalinity_of_ash* feature after applying the transformation.

Figure 4: All the six figures represent the histograms generated by the code with the transformed features.

Percentage of contribution: Janan Jahed: 33% Natalia Gayoso: 33% Andrei Medesan:33%

References

Otten, N. V. (2023). Regression vs classification - understand how to choose and switch between them. *Spot Intelligence*. https://spotintelligence.com/2023/05/02/regression-vs-classification/#:~:text=Classification%5C%20involves%5C%20predicting%5C%20a%5C%20categorical,house%5C%20based%5C%20on%5C%20its%5C%20features.

Solomon, B. (2018). Python histogram plotting: Numpy, matplotlib, pandas seaborn. *Real Python*. https://realpython.com/python-histograms/

University of Groningen. (n.d.-a). Data preprocesing. *This is a notebook provided*.

University of Groningen. (n.d.-b). Logistic regression with python. *This is a notebook provided*.

Yenigün, O. (2022). What are the feature transformation techniques? *Python in Plain English*. https://python.plainenglish.io/what-are-the-feature-transformation-techniques-ba594b523ec4