High Precision Selection of Premium Wines

NAME:	SARTHAK TAYAL
ROLL NO.:	210107076
DATE:	29/03/24
COURSE:	Applications of Artificial Intelligence and Machine Learning in Chemical Engineering (CL-653)

Project Overview:

- *Introduction:*

The project aims to explore the fascinating world of wine quality prediction using Machine Learning. Wine quality is a subjective measure, often influenced by factors such as personal preference, price, and branding. However, there are certain chemical properties of wine that can influence its taste and hence its perceived quality. These include acidity, sugar content, alcohol content, density, and pH among others. The problem we aim to solve is to predict the quality of wine based on these measurable chemical properties. This is an important issue to address as it can help winemakers understand what factors contribute most to wine quality and can guide them in their wine production process to produce higher quality wines.

- *Objectives:*
- **Data Exploration**: To understand the structure, features, and relationships within the dataset. This includes identifying any correlations between the chemical properties of the wine and its quality.
- **Feature Selection**: To identify the most relevant features that have a significant impact on wine quality. This will be achieved through various feature selection techniques.
- **Model Development**: To develop a Machine Learning model that can accurately predict the quality of wine based on its chemical properties. This involves selecting an appropriate algorithm, training the model on the dataset, and tuning it for optimal performance.
- **Model Evaluation:** To evaluate the model's performance using appropriate metrics such as accuracy, precision, recall, and F1 score**.** This will help in understanding how well the model is performing in predicting the wine quality.

Description:

- *Theoretical background:*

1. **Wine Quality Assessment**: Traditionally done by human experts, the assessment of wine quality can be subjective. Using chemical and physical properties for assessment provides a more objective approach.
2. **Chemical Properties of Wine**: The taste and quality of wine are influenced by various chemical properties. Understanding these relationships is key to this project.
3. **Machine Learning for Prediction**: We will use supervised learning techniques to train a model on a dataset of wine properties and their corresponding quality ratings. This model will then predict the quality of new wines.
4. **Model Evaluation**: The performance of the model will be evaluated using a split of the original dataset into a training set and a test set. Metrics such as accuracy, precision, recall, and F1 score will be used for evaluation.

- *Problem Statement:*

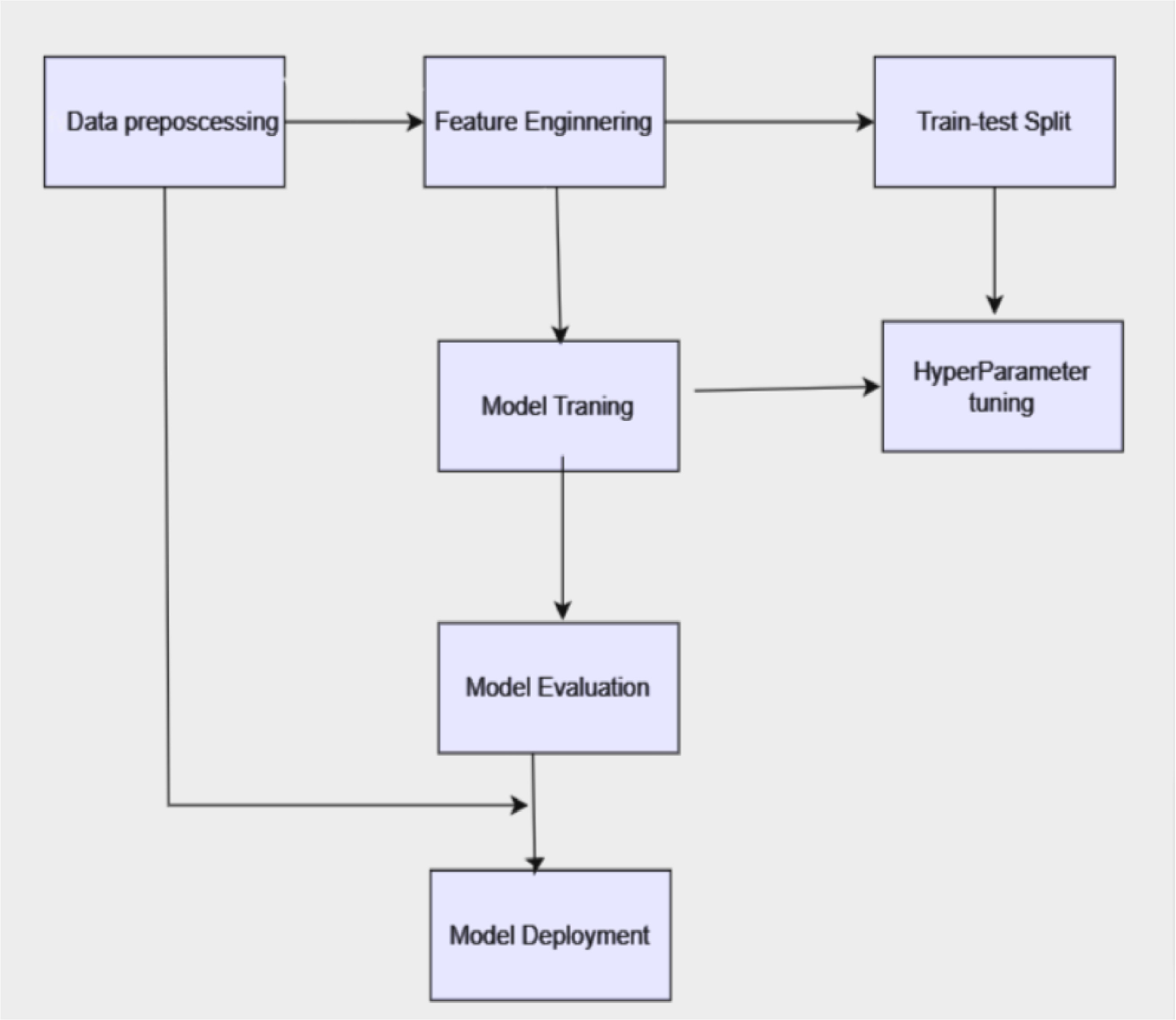
1. **Prediction of Wine Quality**: The primary problem this project aims to solve is predicting the quality of wine based on its chemical properties. This involves creating a model that can accurately determine the quality of a wine given its chemical composition.
2. **Identification of Key Factors**: Another aspect of the problem is identifying which chemical properties have the most significant impact on the quality of the wine. This information can provide valuable insights for winemakers.

3. **Model Performance**: The final part of the problem is ensuring that the model's predictions are reliable and accurate. This involves evaluating the model's performance using appropriate metrics and techniques.

- *Significance of Addressing this Issue:*

1. **Improving Wine Production**: By understanding the chemical properties that significantly influence wine quality, winemakers can adjust their production processes to optimize these factors, potentially improving the quality of their product.
2. **Objective Quality Assessment**: The model provides a more objective and consistent method for assessing wine quality compared to traditional sensory analysis. This can be particularly useful in large-scale wine production and quality control.
3. **Informing Consumer Choice**: The insights gained from this project could also be used to inform consumers about the factors that contribute to wine quality, helping them make more informed purchasing decisions.

Block Diagram:



Data Sources and Data Description:

https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

- *Description:*

The data for this project is obtained from the UCI Machine Learning Repository, specifically the Wine Quality dataset. This dataset is publicly available and can be downloaded directly from the repository's website.

The dataset contains physicochemical properties of red and white variants of the Portuguese "Vinho Verde" wine, along with their corresponding quality ratings. The properties include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content.

- *Data Characteristics:*

1. **Volume**: The Wine Quality dataset is not particularly large compared to many modern datasets. It contains 1,599 instances for red wine and 4,898 instances for white wine, each with 12 attributes. This makes it manageable for most computational resources.
2. **Variety**: The dataset contains a variety of attributes, each representing a different physicochemical property of the wine. These include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. The 'quality' attribute is the target variable and is based on sensory data, while the others are based on physicochemical tests.
3. **Velocity**: In the context of this project, velocity is not a significant factor as the dataset is static. It was collected at one point in time and is not being continuously updated. However, the techniques used in this project could be applied to new data as it becomes available.

- *Nature of Data:*

The Wine Quality dataset is steady state, meaning it was collected at a specific point in time and does not change or update over time.

For this project, this means:

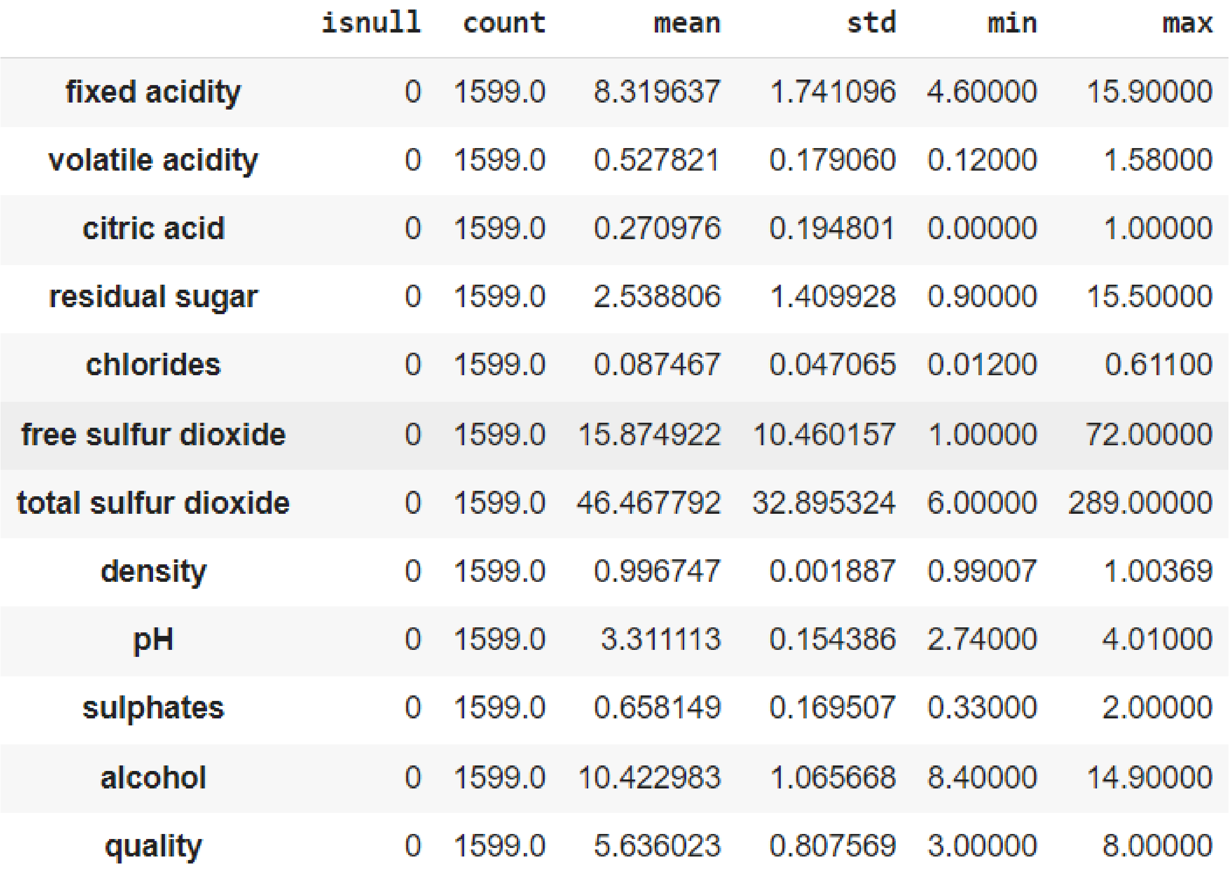
1. We can train our model on the entire dataset without worrying about the model becoming outdated due to new data.
2. We can validate our model's performance using a portion of the dataset.
3. The model's predictions on new data may become less accurate if the underlying patterns in the data change over time, necessitating retraining of the model.

- *Data Preprocessing:*

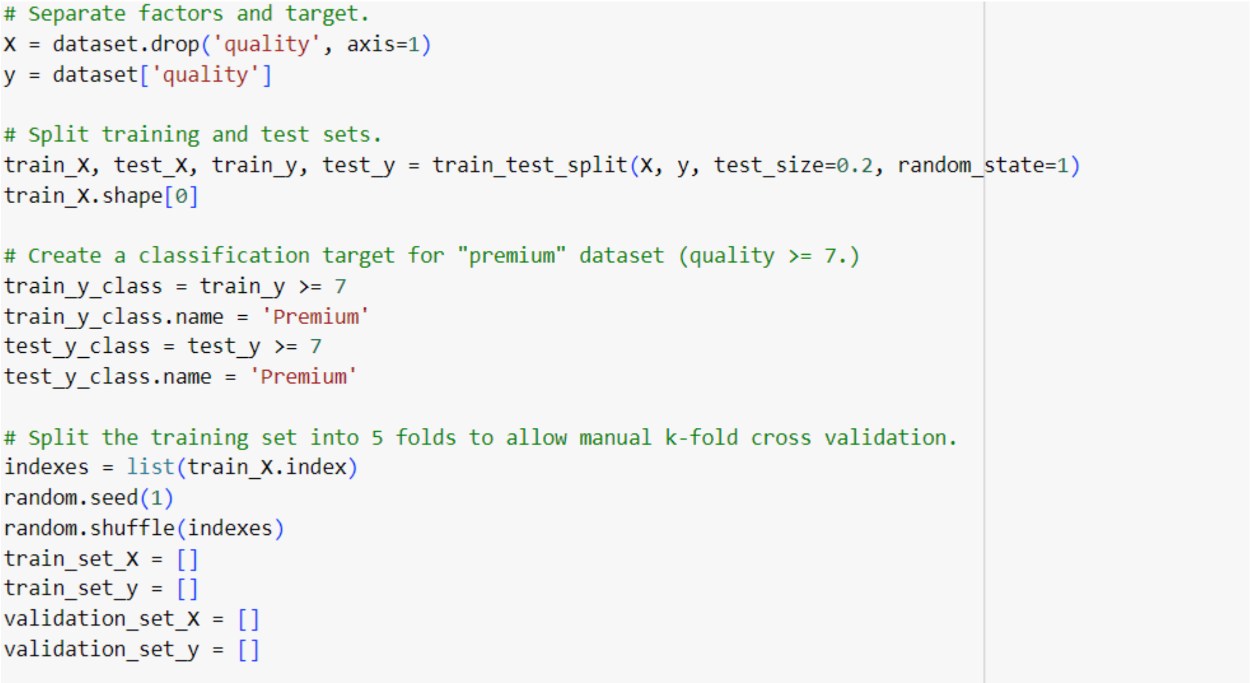
1. **Data Cleaning**: This involves checking for and handling missing values, duplicate entries, and outliers. In the case of missing values, we might decide to fill them using a certain strategy (like mean, median, or mode), or drop the rows or columns with missing values altogether, depending on the extent of the missing data. Outliers can be detected using various methods and might be capped, winsorized, or removed, depending on the situation.
2. **Data Transformation**: This could involve creating new features that might be useful for our analysis. For example, we might want to create a binary 'good quality' feature based on the 'quality' feature.
3. **Data Normalization**: Many machine learning algorithms perform better when numerical input variables are scaled to a standard range. This might involve standardization (shifting the distribution to have a mean of

zero and a standard deviation of one) or normalization (scaling the distribution to have a range between 0 and 1).

4. **Encoding Categorical Variables**: If the dataset contains categorical variables, these would need to be converted into a format that can be understood by our machine learning algorithms. This might involve one-hot encoding or ordinal encoding, depending on the nature of the categorical variable.
5. **Data Splitting**: The data will need to be split into a training set and a test set. The model will be trained on the training set and its performance evaluated on the test set.



	isnull	count	mean	std	min	max
fixed acidity	0	1599.0	8.319637	1.741096	4.60000	15.90000
volatile acidity	0	1599.0	0.527821	0.179060	0.12000	1.58000
citric acid	0	1599.0	0.270976	0.194801	0.00000	1.00000
residual sugar	0	1599.0	2.538806	1.409928	0.90000	15.50000
chlorides	0	1599.0	0.087467	0.047065	0.01200	0.61100
free sulfur dioxide	0	1599.0	15.874922	10.460157	1.00000	72.00000
total sulfur dioxide	0	1599.0	46.467792	32.895324	6.00000	289.00000
density	0	1599.0	0.996747	0.001887	0.99007	1.00369
pH	0	1599.0	3.311113	0.154386	2.74000	4.01000
sulphates	0	1599.0	0.658149	0.169507	0.33000	2.00000
alcohol	0	1599.0	10.422983	1.065668	8.40000	14.90000
quality	0	1599.0	5.636023	0.807569	3.00000	8.00000



```
# Separate factors and target.
X = dataset.drop('quality', axis=1)
y = dataset['quality']

# Split training and test sets.
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=1)
train_X.shape[0]

# Create a classification target for "premium" dataset (quality >= 7.)
train_y_class = train_y >= 7
train_y_class.name = 'Premium'
test_y_class = test_y >= 7
test_y_class.name = 'Premium'

# Split the training set into 5 folds to allow manual k-fold cross validation.
indexes = list(train_X.index)
random.seed(1)
random.shuffle(indexes)
train_set_X = []
train_set_y = []
validation_set_X = []
validation_set_y = []
```

Strategies for AI/ML Model Development:

- *Model Development:*

For the Wine Quality dataset, we're considering the following models due to their suitability for regression problems:

1. **Linear Regression**
2. **Logistic Regression**
3. **Random Forest Regressor**
4. **XGBoost Classification**
5. **XGBoost Regression**
6. **Random Forest Classification**

- *Training:*

Our approach to model training includes:

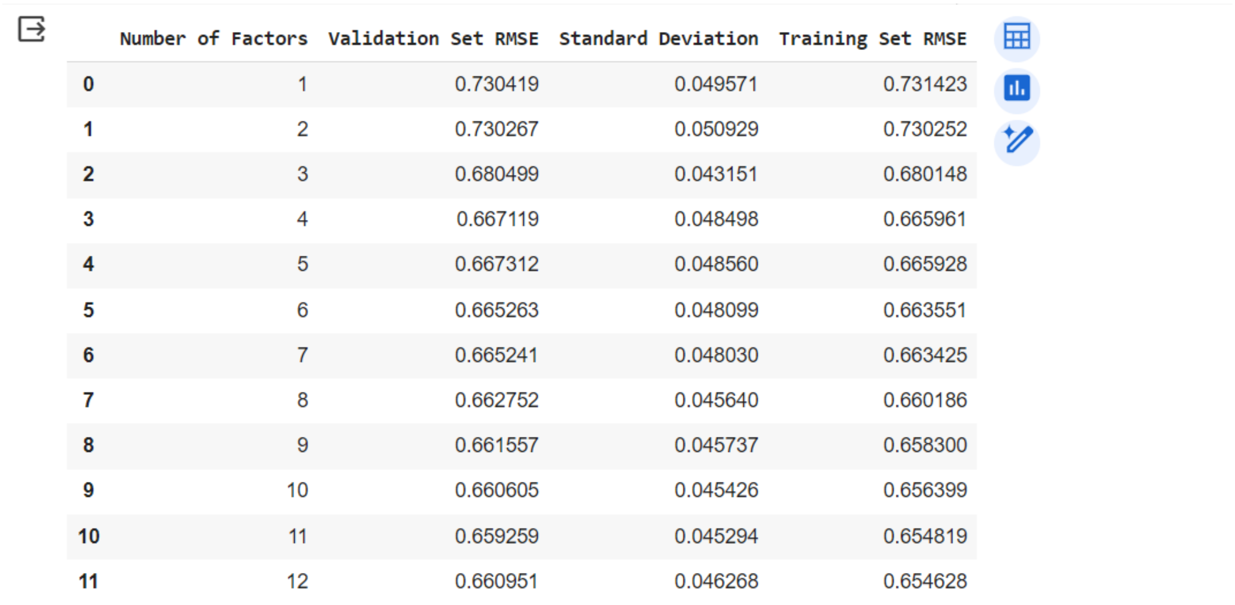
1. **Data Preprocessing**: Cleaning, handling missing values and outliers, normalizing numerical features, and encoding categorical features.
2. **Feature Selection**: Identifying important features using correlation analysis or feature importance.
3. **Model Training**: Training various models on a split training set, starting with simpler models and moving to complex ones if needed.
4. **Model Evaluation**: Evaluating models on a test set using metrics like Mean Absolute Error or R-squared.
5. **Model Optimization**: Fine-tuning the best model using grid search or random search for hyperparameter optimization.

6. **Validation**: Validating the final model using cross-validation to ensure robust performance.

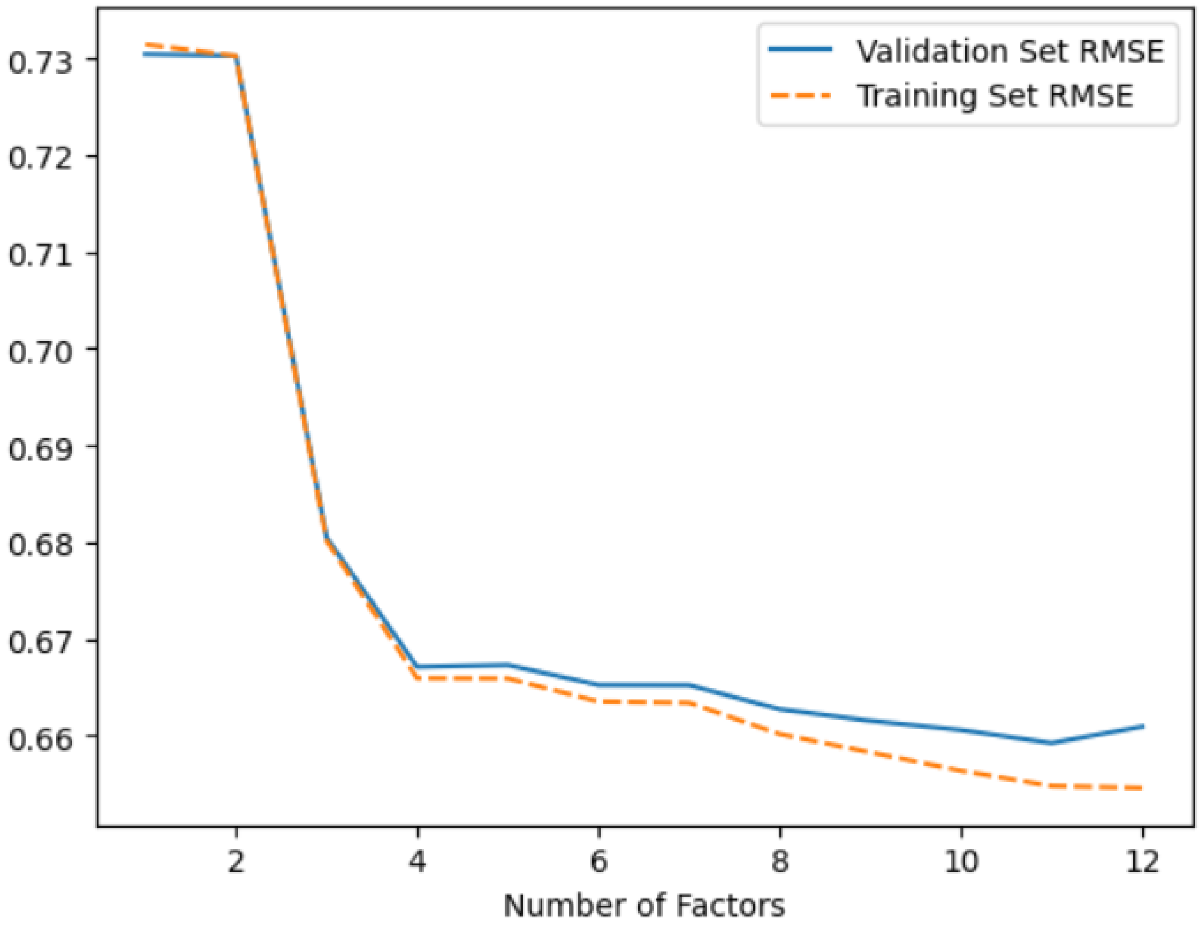
- *Evaluation and Validation:*

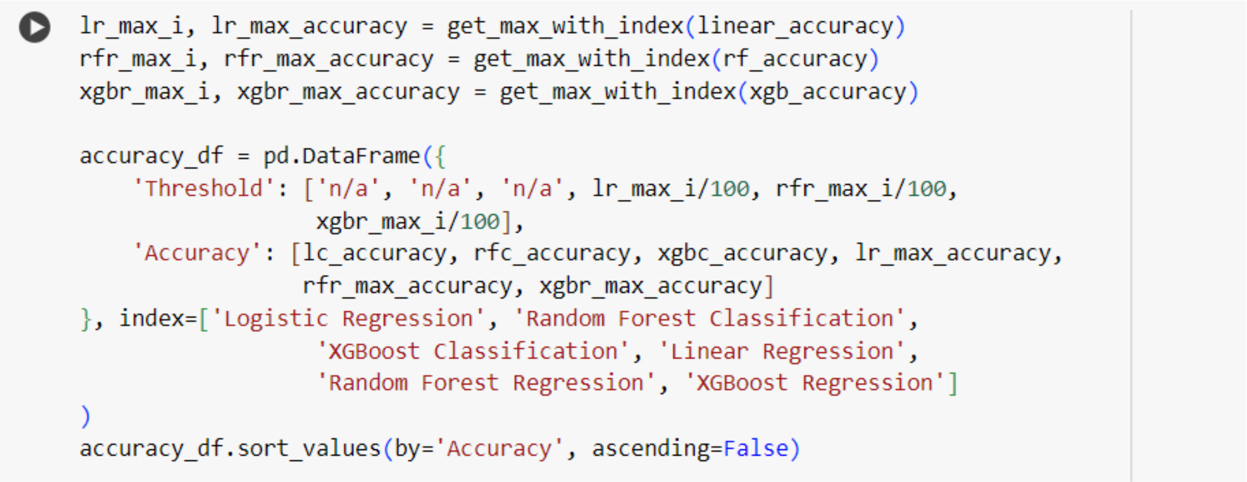
Evaluation Metrics: We'll use Mean Absolute Error (MAE) for its interpretability, Mean Squared Error (MSE) for its sensitivity to large errors, and R-squared (R^2) to measure the overall model fit, providing a comprehensive evaluation of our model's performance.

Validation Strategy: We'll validate our model using k-fold cross-validation for robustness and, if available, external validation on unseen datasets for generalizability.



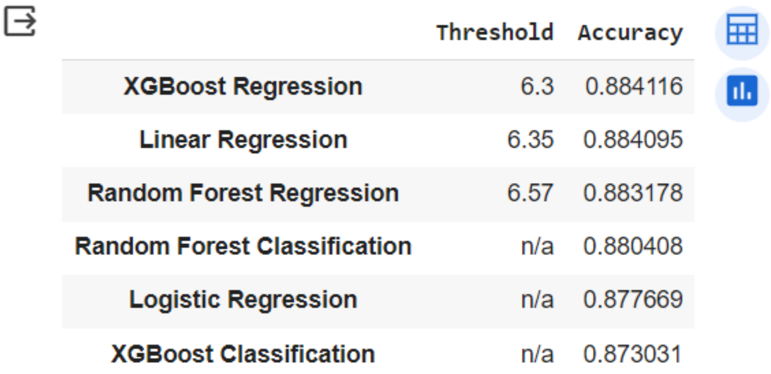
	Number of Factors	Validation Set RMSE	Standard Deviation	Training Set RMSE
0	1	0.730419	0.049571	0.731423
1	2	0.730267	0.050929	0.730252
2	3	0.680499	0.043151	0.680148
3	4	0.667119	0.048498	0.665961
4	5	0.667312	0.048560	0.665928
5	6	0.665263	0.048099	0.663551
6	7	0.665241	0.048030	0.663425
7	8	0.662752	0.045640	0.660186
8	9	0.661557	0.045737	0.658300
9	10	0.660605	0.045426	0.656399
10	11	0.659259	0.045294	0.654819
11	12	0.660951	0.046268	0.654628


0.73
0.72
0.71
0.70
0.69
0.68
0.67
0.66
2
4
6
8
10
12
Number of Factors
Validation Set RMSE
Training Set RMSE

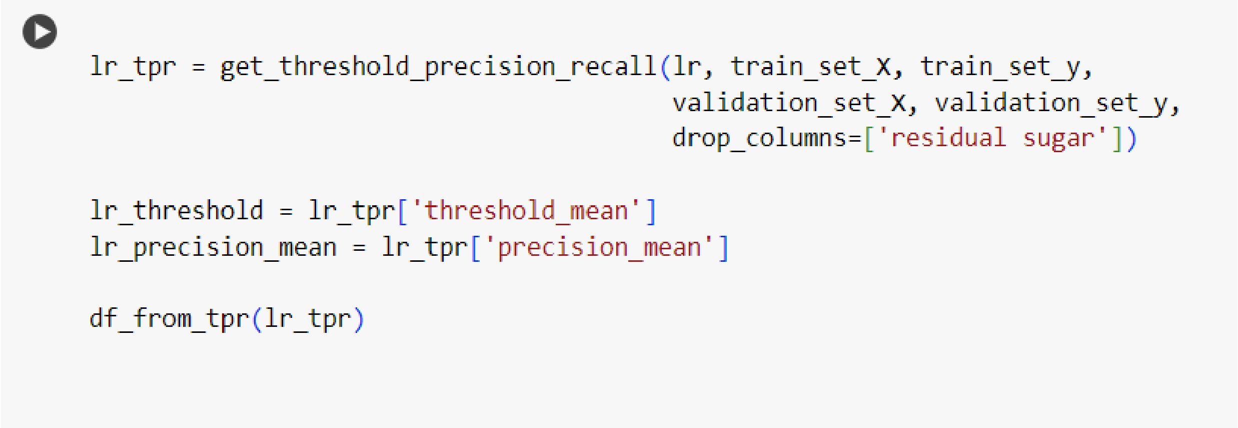


```python
lr_max_i, lr_max_accuracy = get_max_with_index(linear_accuracy)
rfr_max_i, rfr_max_accuracy = get_max_with_index(rf_accuracy)
xgbr_max_i, xgbr_max_accuracy = get_max_with_index(xgb_accuracy)

accuracy_df = pd.DataFrame({
    'Threshold': ['n/a', 'n/a', 'n/a', lr_max_i/100, rfr_max_i/100,
                  xgbr_max_i/100],
    'Accuracy': [lc_accuracy, rfc_accuracy, xgbc_accuracy, lr_max_accuracy,
                 rfr_max_accuracy, xgbr_max_accuracy]
}, index=['Logistic Regression', 'Random Forest Classification',
          'XGBoost Classification', 'Linear Regression',
          'Random Forest Regression', 'XGBoost Regression']
)
accuracy_df.sort_values(by='Accuracy', ascending=False)
```



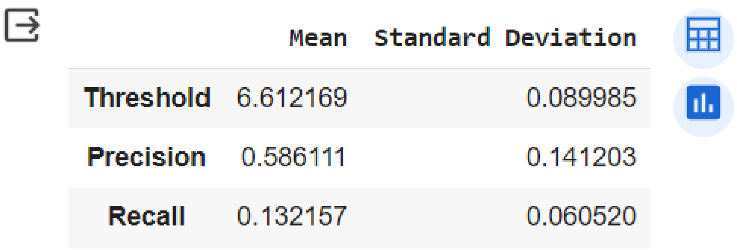
	Threshold	Accuracy
XGBoost Regression	6.3	0.884116
Linear Regression	6.35	0.884095
Random Forest Regression	6.57	0.883178
Random Forest Classification	n/a	0.880408
Logistic Regression	n/a	0.877669
XGBoost Classification	n/a	0.873031



```python
lr_tpr = get_threshold_precision_recall(lr, train_set_X, train_set_y,
                                        validation_set_X, validation_set_y,
                                        drop_columns=['residual sugar'])

lr_threshold = lr_tpr['threshold_mean']
lr_precision_mean = lr_tpr['precision_mean']

df_from_tpr(lr_tpr)
```

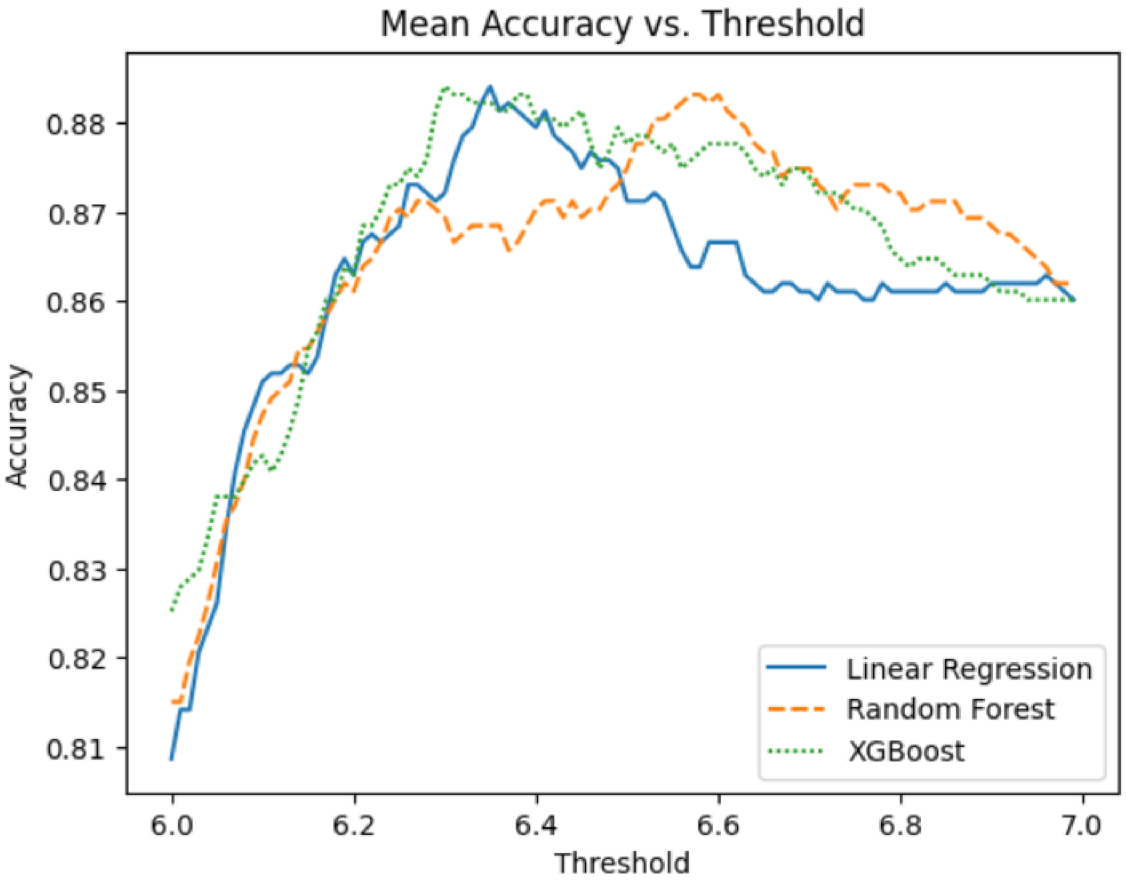


	Mean	Standard Deviation
Threshold	6.612169	0.089985
Precision	0.586111	0.141203
Recall	0.132157	0.060520

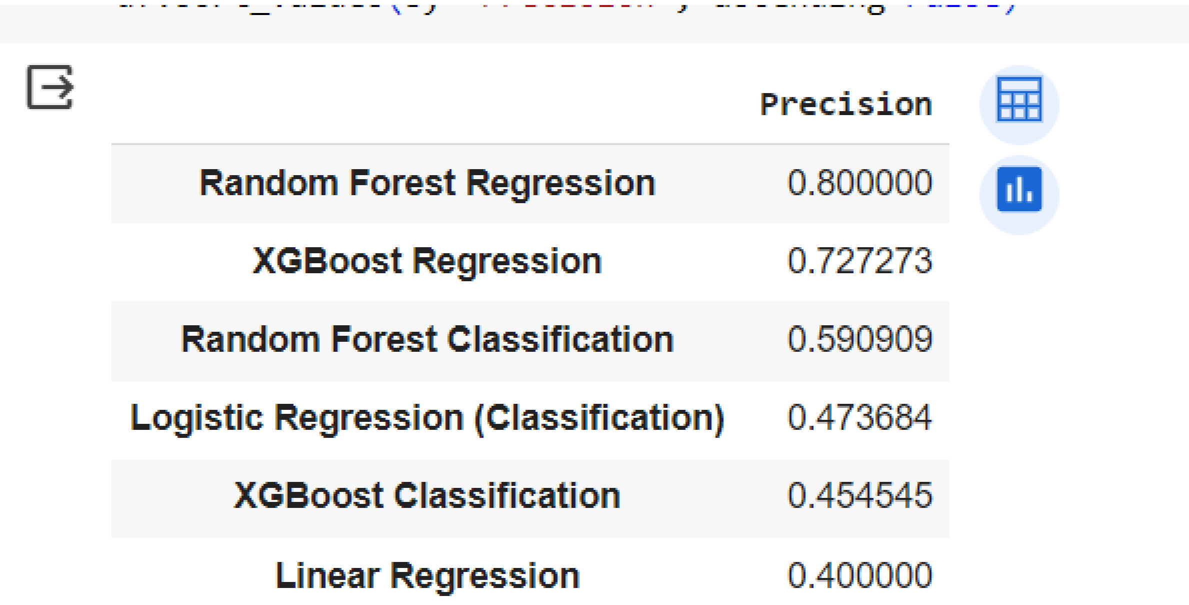
```
xgbr_tpr = get_threshold_precision_recall(xgbr, train_set_X, train_set_y,
                                          validation_set_X, validation_set_y,
                                          drop_columns=['residual sugar'])
xgbr_threshold = xgbr_tpr['threshold_mean']
xgbr_precision_mean = xgbr_tpr['precision_mean']

df_from_tpr(xgbr_tpr)
```

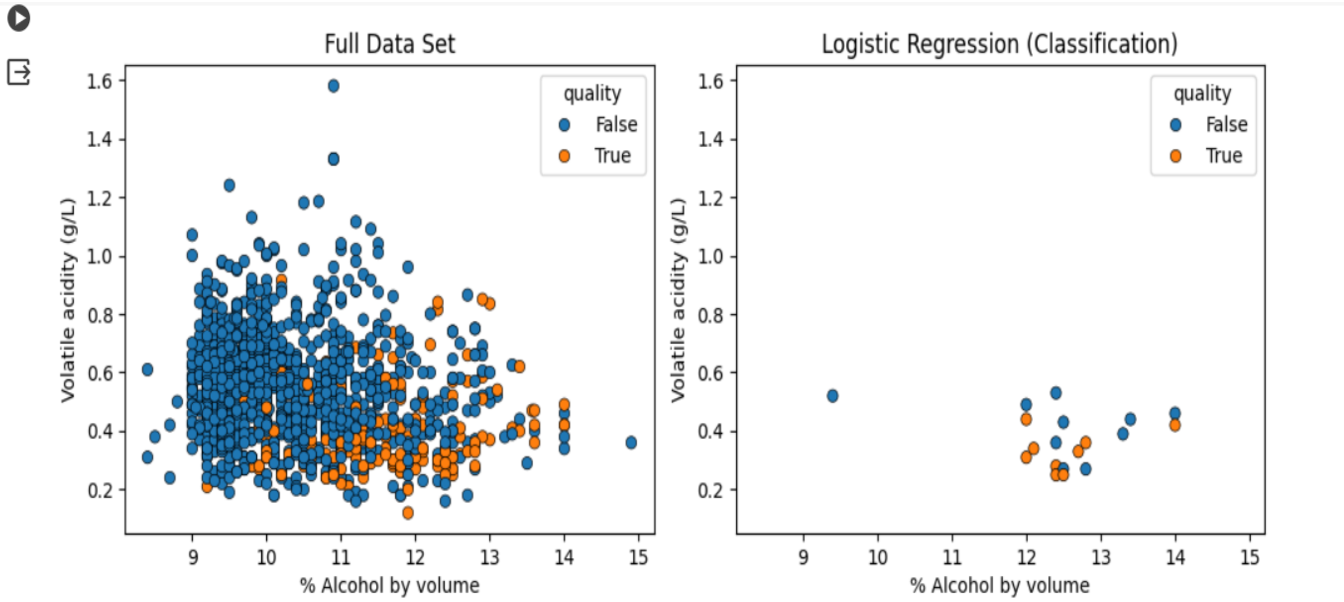
	Mean	Standard Deviation
Threshold	6.640766	0.060236
Precision	0.755556	0.167037
Recall	0.165332	0.050306

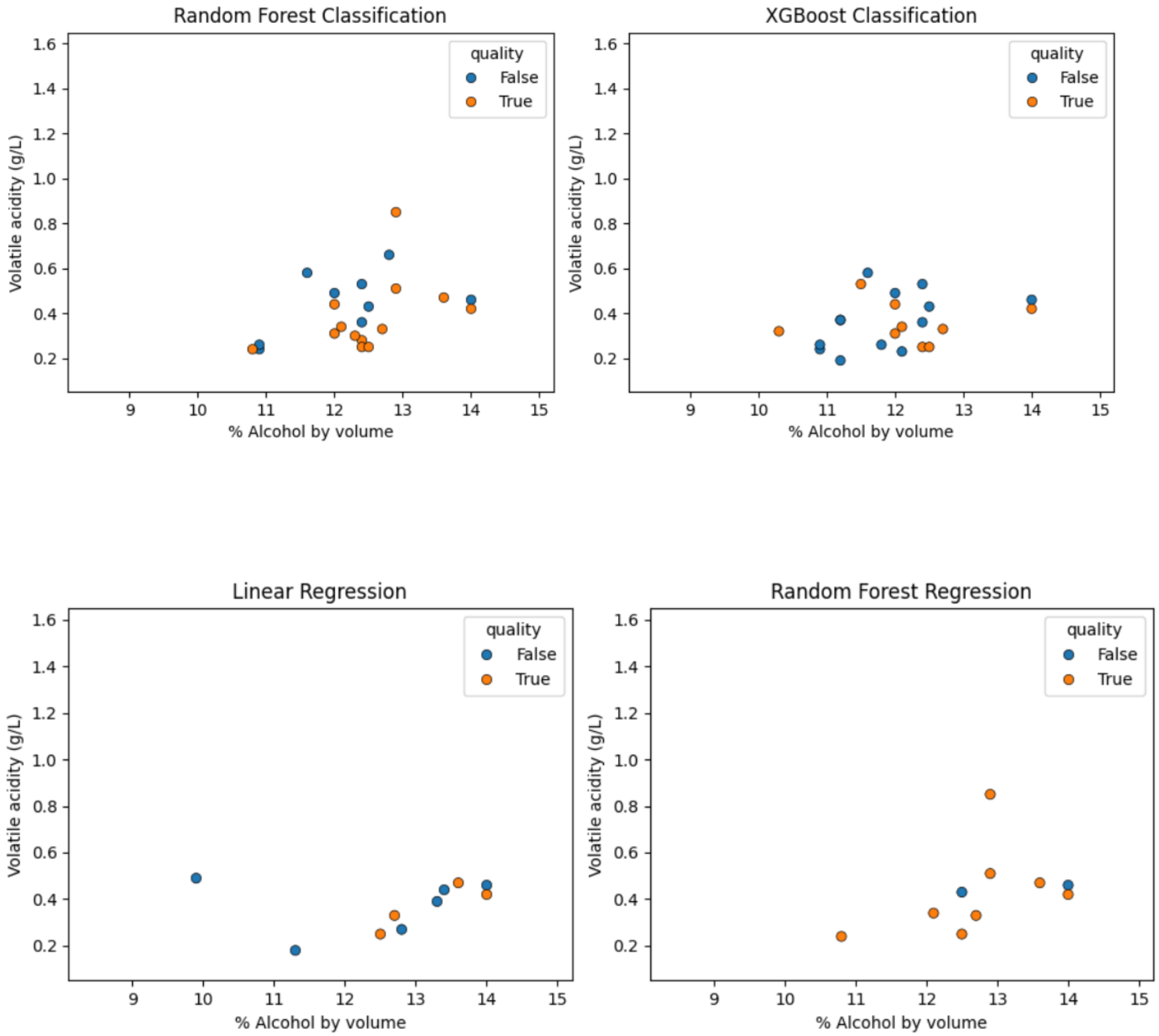


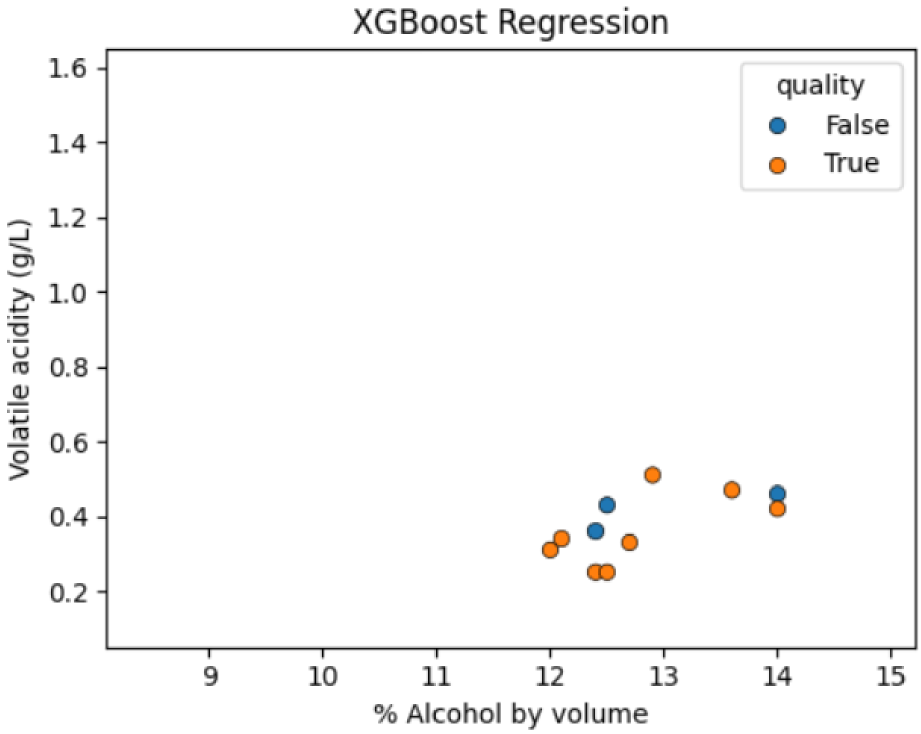
Results :

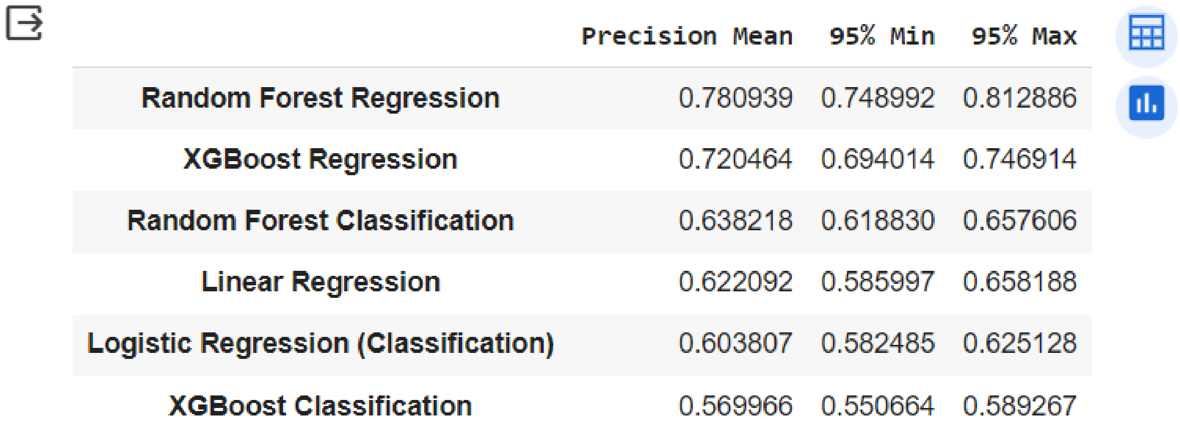


	Precision
Random Forest Regression	0.800000
XGBoost Regression	0.727273
Random Forest Classification	0.590909
Logistic Regression (Classification)	0.473684
XGBoost Classification	0.454545
Linear Regression	0.400000

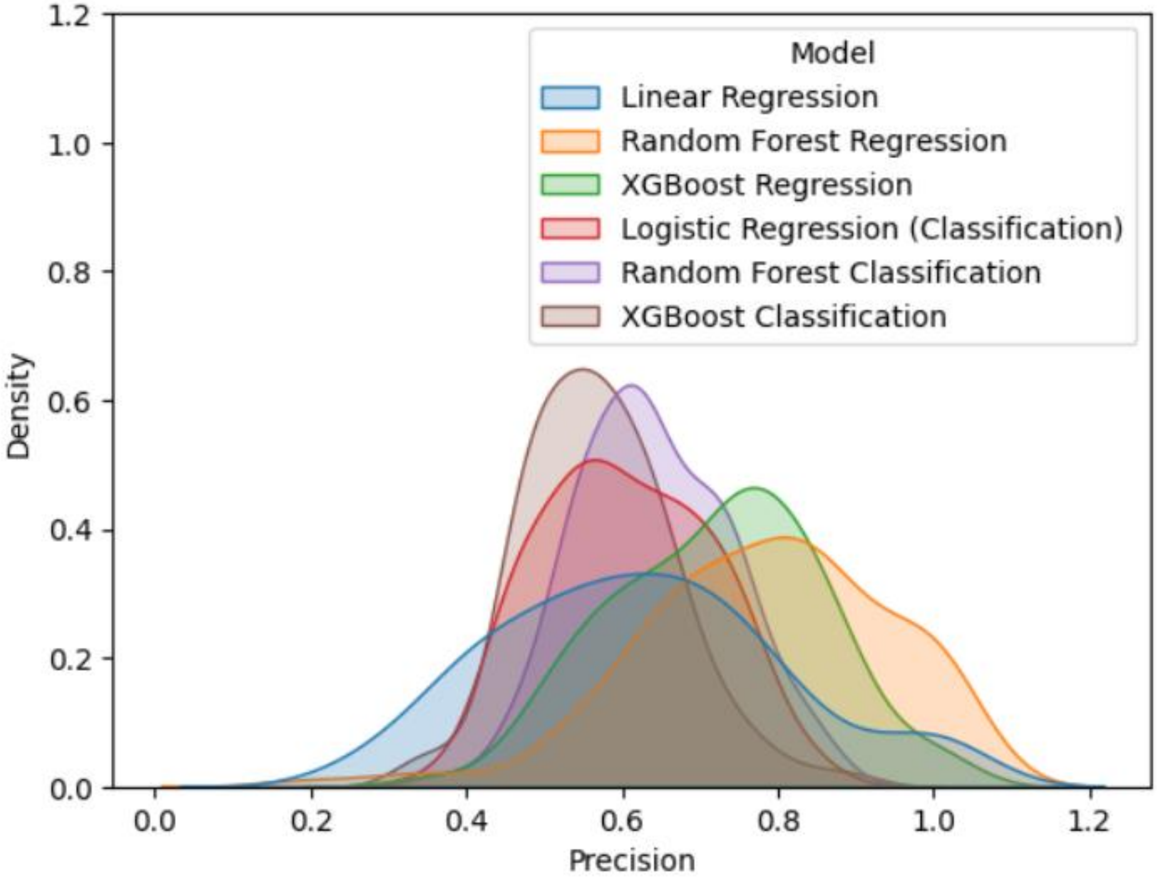



Random Forest Classification
XGBoost Classification
Linear Regression
Random Forest Regression
quality
False
True
Volatile acidity (g/L)
% Alcohol by volume
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
9
10
11
12
13
14
15





	Precision Mean	95% Min	95% Max
Random Forest Regression	0.780939	0.748992	0.812886
XGBoost Regression	0.720464	0.694014	0.746914
Random Forest Classification	0.638218	0.618830	0.657606
Linear Regression	0.622092	0.585997	0.658188
Logistic Regression (Classification)	0.603807	0.582485	0.625128
XGBoost Classification	0.569966	0.550664	0.589267


Model
Linear Regression
Random Forest Regression
XGBoost Regression
Logistic Regression (Classification)
Random Forest Classification
XGBoost Classification
Density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Precision