Predicting Portugal white wine quality

by Kittipong Wongwipasamitkun, Nicole Tu, Sho Inagaki 2023/11/19

In [1]:
```
import numpy as np
import pandas as pd
import requests
import zipfile
import altair as alt
from sklearn import set_config
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.compose import make_column_transformer, make_column_selector
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import fbeta_score, make_scorer
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
from scipy.stats import uniform
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

Summary

We tried to make the classification model using the Polynomial Regression with Ridge Regularization algorithm with Randomized Search Hyperparameters which can predict Portugal white wine quality rating (on scale 0-10) through the physicochemical properties of the test wine. The model has trained on the Portugal white wine data set with 4898 observations. In the conclusion, the model performance is not quite good enough both on training data and on an unseen test data set with the test score at around 0.32 with the average train R^2 at 0.35 and the average test R^2 at 0.27 also with high root MSE and MSE (Mean Squared Error).

The reason we suspect the model cannot predict well is that the wine quality can be judge widely and vary depends on each individual preference taste. Moreover, there is no standard on the taste, for example, high or low in acidity or alcohol level or sulfur level cannot indicate the wine is in good quality or not (It can be both ways!!). As such, we believe this model is at, or close to, the starter required for studying further and could run more collected data to analyze the combination of physicochemical properties which will announce quality of the wine, although more researches need to improve the model performance and understand the characteristics of incorrectly predicted pattern would be in need to investigate further.

Introduction

Referring to WSET (the Wine & Spirits Education Trust), Wine tasting notes can be described by Systematic Approach to Tasting (SAT) (https://www.wsetglobal.com/media/13271/wset_l4wines_sat_en_aug2023.pdf) , which is consisted of

1. Appearance (colour, clarity, intensity, and other observations)
2. Nose (condition, intensity, aroma characteristics, aroma development)
3. Palate (sweetness, acidity, tannin, alcohol, body, flavour intensity, flavour characteristics, finish, and other observations)
4. Conclusion (quality, readiness for drinking and potential for ageing).

The wine quality which is in the conclusion part, consists mainly on Balance, Length, Intensity and Complexity (BLIC). The result of these qualities, all came from the chemical components in the wine. Subsequently, nowadays, the quality of wine can be determined roughly from the physicochemical components of the wine.

As the physicochemical properties have been related to the wine quality, so we aim to create a machine learning algorithm to predict the quality of wine from the measurement of physicochemical values. Answering this question can help both customers and winemakers to screen or adjust or make decision to the prior wine quality rating derived from the model according to its physicochemical values.

This machine learning algorithm will aim to study only the Portugal white wine to reduce biased from the types of wine and the origin sources of wine as the start point to assess the quality of wine via its physicochemical features.

Methods

Data

This data set used in this project is related to white vinho verde wine samples from the north of Portugal created By P. Cortez, A. Cerdeira, Fernando Almeida, Telmo Matos, J. Reis. 2009. The dataset was sourced from website for downloading these datasets is the UC Irvine Machine Learning Repository (https://archive.ics.uci.edu/dataset/186/wine+quality). In addition, these datasets stored the physicochemical properties data on wines and the quality rating to compare and make the quality prediction model.

Analysis

As we have many variables to concern and most of them have correlation to the target quality, which it is best to not drop these correlated features. Also in the real world with multiple explanatory variables, it is less likely for model to have a linear pattern, and we believe that polynomial regression can be a more realistic model to predict wine quality. The reason we use polynomial regression rather than a linear regression is that it allows us to capture more complex relationships between the predictors and the target variable. By introducing polynomial terms, we can account for non-linear patterns that might exist in the data, enabling the model to better fit the intricacies of wine quality prediction. Moreover, polynomial regression provides greater flexibility, allowing us to uncover potential curvilinear associations between features and the quality of wine, which a linear regression might overlook. This approach enhances our ability to create a more nuanced and accurate predictive model for wine quality assessment. Therefore, we choose the **Polynomial regression with ridge regularization** to reduce the effect of multicolinearity, and use **Random search** to optimize the hyperparameters. Data was partitioned by 70% for the training set and 30% for the test set. All variables were standardized just prior to model fitting. The Python programming language (Van Rossum and Drake 2009) and the following Python packages were used to perform the analysis: numpy(Harris et al. 2020), Pandas (McKinney 2010), altair (VanderPlas, 2018), scikit-learn (Pedregosa et al. 2011). The code used to perform the analysis and create this report can be found here: https://github.com/UBC-MDS/DSCI_522_group16/tree/main/src/portugal_wine_quality_predictor.ipynb

Discussion

We tried to find out whether each of the physicochemical properties might be useful to predict the wine quality rating. We examined the data set that it doesn't have Null value or any adjustment needed for any values. After that we make a correlation matrix for all the variables in this data set to recognize the pattern or choose to drop some variable out.

In [2]:
```
# download data as zip and extract

# If you want to download file from url then uncomment below

# url = "https://archive.ics.uci.edu/dataset/186/wine+quality/wine+quality.zip"
# request = requests.get(url)
# with open("../data/raw/wine+quality.zip", 'wb') as f:
#     f.write(request.content)

# with zipfile.ZipFile("../data/raw/wine+quality.zip", 'r') as zip_ref:
#     zip_ref.extractall("../data/raw")
```

In [3]:
```
# pre-process data (e.g., drop na, check data info)
# read white wine data
white_wine = pd.read_csv('../data/Raw/winequality-white.csv', sep=';')
white_wine.dropna(inplace=True)
white_wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

In [4]:
```
np.random.seed(522)
# 1. split train and test data as df, save to folder.
white_train, white_test = train_test_split(white_wine, train_size=0.70, random_stat
white_train.to_csv("../data/Processed/white_train.csv")
white_test.to_csv("../data/Processed/white_test.csv")
```

In [5]:
```
# 2. EDA by visualization

white_train.head()

# Visualize correlation
white_train.corr().style.background_gradient()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	den
fixed acidity	1.000000	-0.022159	0.283248	0.098448	0.017915	-0.051813	0.090352	0.26
volatile acidity	-0.022159	1.000000	-0.147688	0.064137	0.076459	-0.088009	0.085720	0.02
citric acid	0.283248	-0.147688	1.000000	0.107066	0.127680	0.092379	0.118290	0.16
residual sugar	0.098448	0.064137	0.107066	1.000000	0.076111	0.280611	0.396009	0.84
chlorides	0.017915	0.076459	0.127680	0.076111	1.000000	0.099581	0.192030	0.24
free sulfur dioxide	-0.051813	-0.088009	0.092379	0.280611	0.099581	1.000000	0.620482	0.28
total sulfur dioxide	0.090352	0.085720	0.118290	0.396009	0.192030	0.620482	1.000000	0.52
density	0.269401	0.029973	0.160669	0.843845	0.241153	0.282654	0.526727	1.00
pH	-0.423450	-0.029339	-0.178773	-0.198103	-0.097633	0.021046	0.014098	-0.09
sulphates	-0.024174	-0.050048	0.064732	-0.029428	0.016289	0.067162	0.135311	0.07
alcohol	-0.125424	0.062962	-0.093406	-0.457366	-0.349867	-0.244811	-0.456719	-0.77
quality	-0.118615	-0.195553	-0.018860	-0.085984	-0.200642	0.008141	-0.175026	-0.29

Figure 1. Correlation matrix indicating potential multicolinarity among features.

From figure 1, it turns out that among explanatory features, density and residual sugar are highly correlated (0.844); density and total sulfur dioxide are highly correlated(0.527); total sulfur dioxide and free sulfur dioxide are highly correlated (0.620).

Since the free SO2 is the active, unbound form that contributes to antioxidant and antimicrobial properties, and the Total SO2 includes both free and bound forms, providing an overall measure of sulfur dioxide content in the win, we drop the free sulfur dioxide from the data. .

In [6]:
```
# drop redundant feature
white_train = white_train.drop(columns=["free sulfur dioxide"])
```

Let's examine the relationship between the three highly correlated features with target variable by scatter plot matrix, figure 2, as shown below:

In [7]:
```
alt.data_transformers.enable('vegafusion')  # simplify working with large datasets

alt.Chart(white_train).mark_point(opacity=0.3, size=10).encode(
     alt.X(alt.repeat('row'), type='quantitative'),          #encodes the x-axis and
     alt.Y(alt.repeat('column'), type='quantitative')
).properties(
    width=100,
    height=100
).repeat(
    column=['density', 'residual sugar', 'total sulfur dioxide', 'quality'],
    row=['density', 'residual sugar', 'total sulfur dioxide','quality'])
```

Out[7]:

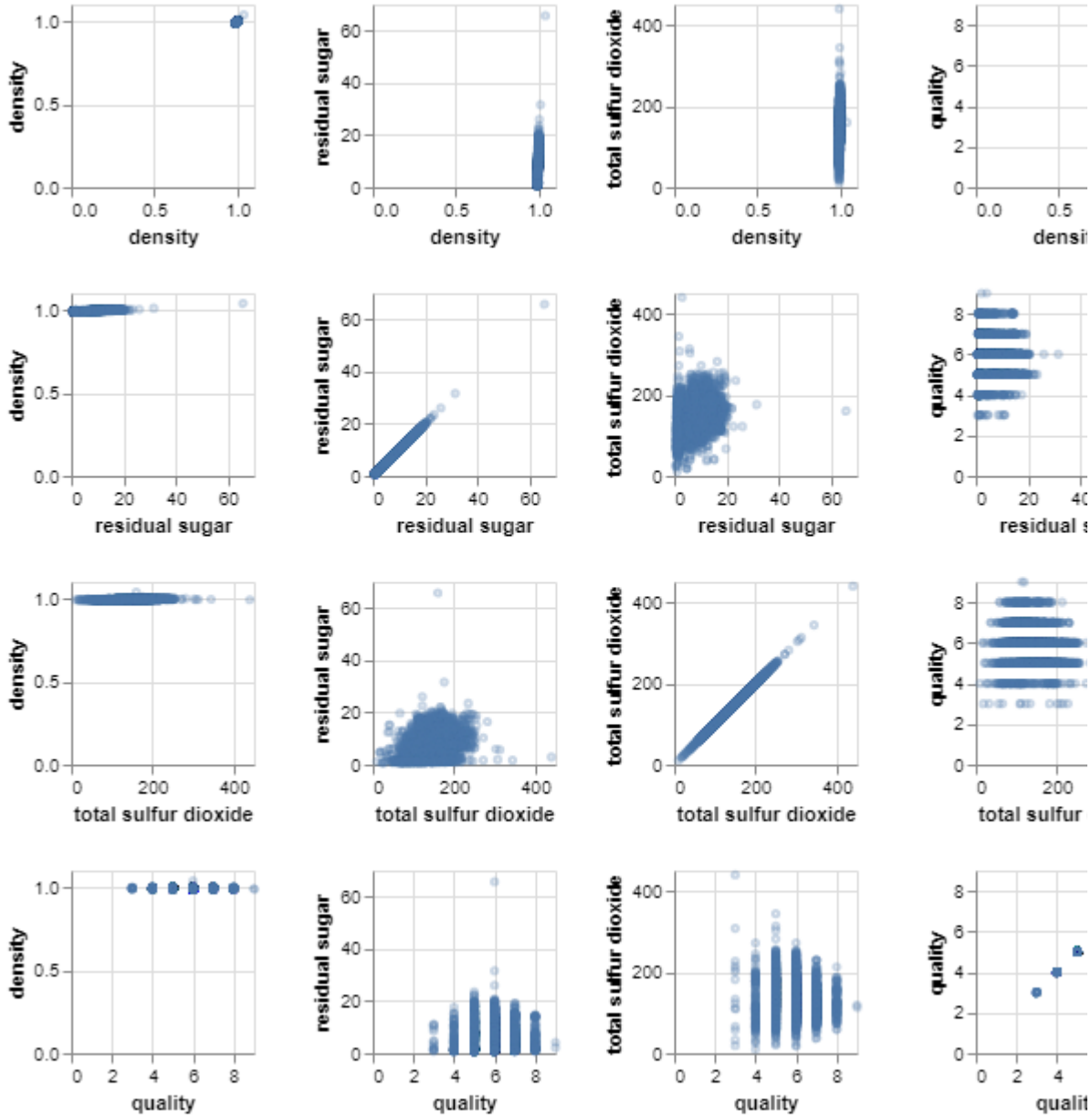


Figure 2. Scatter Plot Matrix indicating correlation between three highly correlated features with target variable

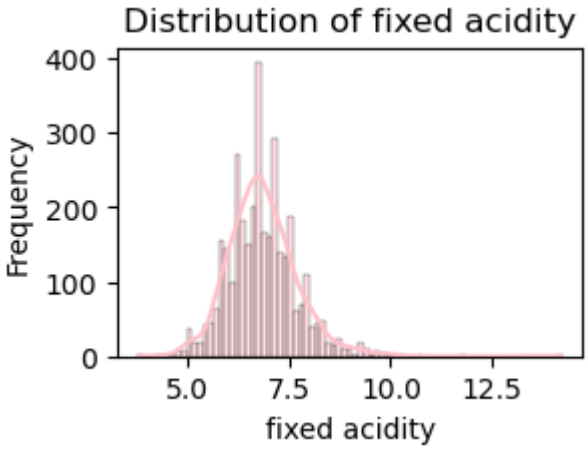
From figure 2, since both residual sugar, density, and total sulfur dioxide have correlation with the target quality, it is not a good idea to drop these correlated features. Instead, we

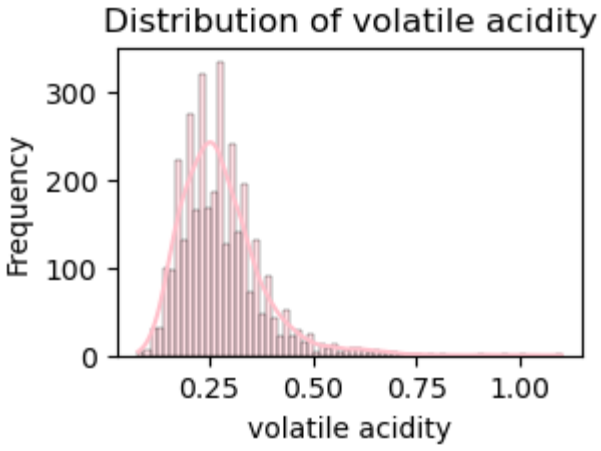
pick the **polynomial regression with ridge regularization** to reduce multicolinearity, and use **Random search** to optimize the hyperparamters.

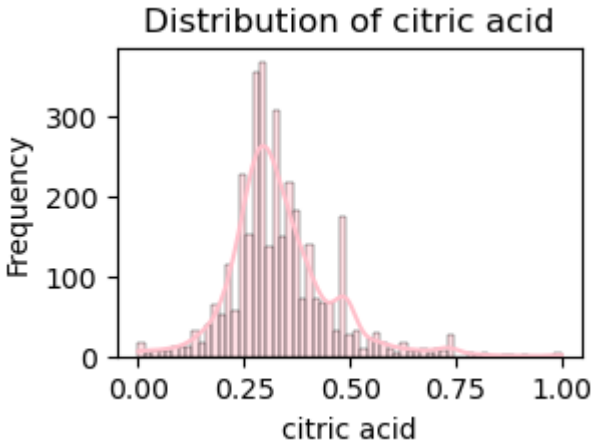
The reason we use polynomial regression rather than a linear regression is that in the real world scenario with multiple explanatory variables, it is less likely for model to have a linear pattern, and we believe that polynomial regression can be a more realistic model to predict wine quality.

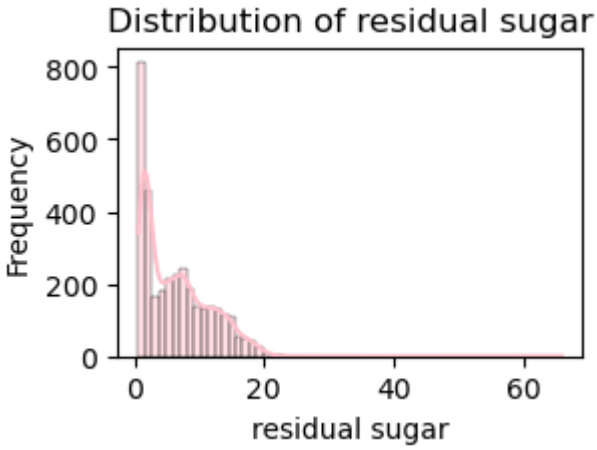
Now lets examine the distribution of the features to decide how to preprocess them:

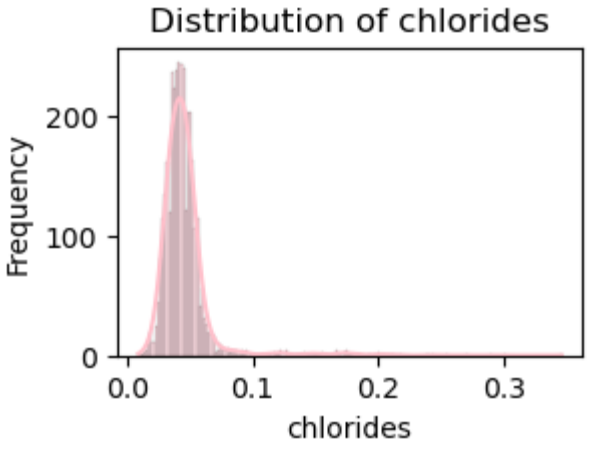
In [8]:
```
# Create a histogram for each column
for column in white_train.columns:
    plt.figure(figsize=(3, 2))
    sns.histplot(white_train[column], kde=True, color='pink')
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

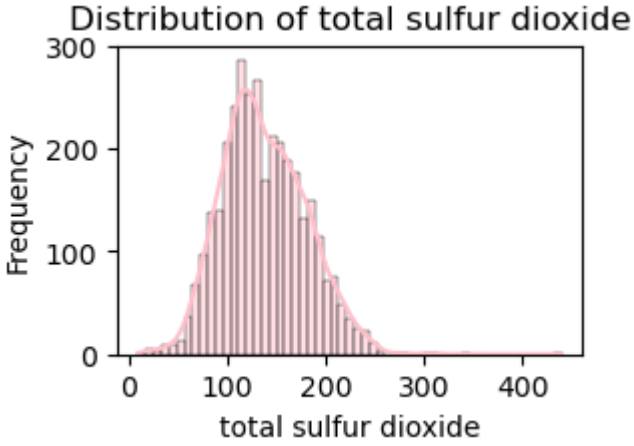


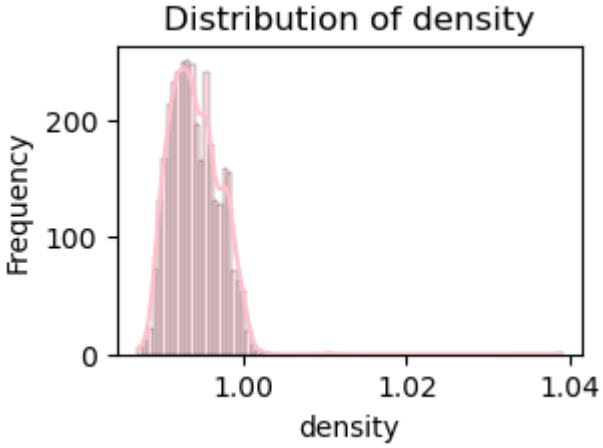


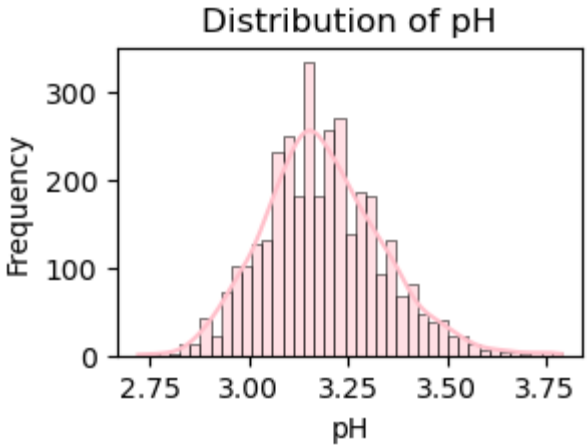

Distribution of citric acid
Frequency
300
200
100
0
0.00
0.25
0.50
0.75
1.00
citric acid


Distribution of residual sugar
Frequency
800
600
400
200
0
0
20
40
60
residual sugar


Distribution of chlorides
Frequency
200
100
0
0.0
0.1
0.2
0.3
chlorides


Distribution of total sulfur dioxide
Frequency
300
200
100
0
0
100
200
300
400
total sulfur dioxide


Distribution of density
Frequency
200
100
0
1.00
1.02
1.04
density


Distribution of pH
Frequency
300
200
100
0
2.75
3.00
3.25
3.50
3.75
pH

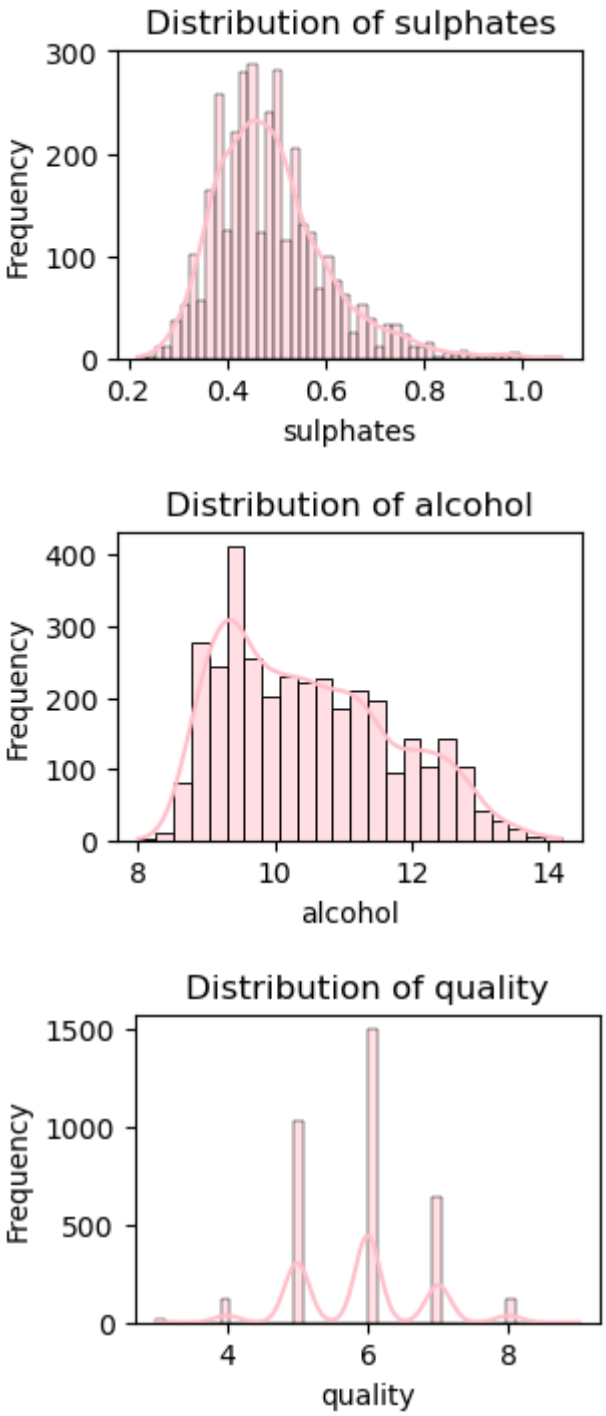


Figure 3. Distribution of features in training set.

From figure 3, most of the features follow an approximately normal distribution, while the residual sugar is slightly skewed. Given the distribution of the features, we preprocess the data by **standardization and imputation with median value**.

In [9]:
```
x_train_w, y_train_w = white_train.drop(columns=["quality"]), white_train["quality"
x_test_w, y_test_w = white_test.drop(columns=["quality"]), white_test["quality"]
```

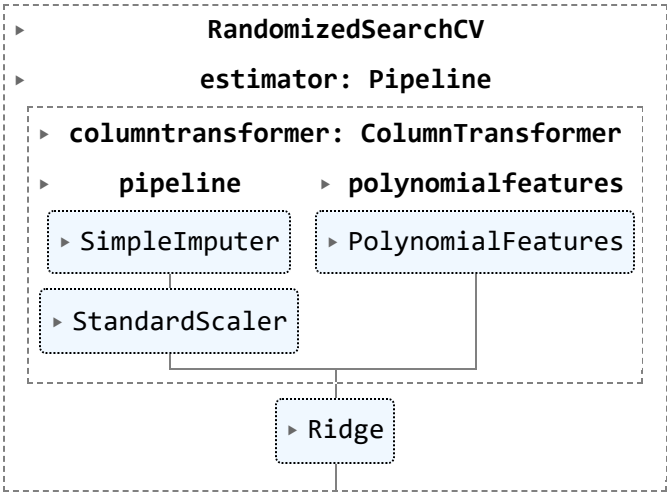
In [10]:
```
# Polinomial regression
numeric_transformer = make_pipeline(SimpleImputer(strategy="median"), StandardScale
```

```
numeric_feats = x_train_w.columns.tolist()

# Create a ColumnTransformer
ct = make_column_transformer(
    (numeric_transformer, numeric_feats),  # Apply numeric transformer to numeric_f
    (PolynomialFeatures(), numeric_feats) # Apply poly_transformer to numeric_feats
  )

Pipe_poly=make_pipeline(ct, Ridge())
```

In [11]:
```
# Hyperparameter Tuning by Random Search
param_dist = { #can pass either a distribution or list of values for random search
    "columntransformer__polynomialfeatures__degree":list(range(1, 8)),
    "ridge__alpha": np.logspace(-2, 2, 15)}

random_search = RandomizedSearchCV(Pipe_poly, param_distributions = param_dist, n_i
n_jobs=-1, return_train_score=True)
```

In [12]:
```
# fit the random search with training data
random_search.fit(x_train_w, y_train_w)
```

Out[12]:



In [13]:
```
random_search.best_params_
```

Out[13]:
```
{'ridge__alpha': 0.13894954943731375,
 'columntransformer__polynomialfeatures__degree': 2}
```

From above, we got best hyperparameters from random search, which are ridge alpha around 0.13895 and the polynomial degree of 2

In [14]:
```
best_model = random_search.best_estimator_ # Get the best model based on optimizati
```

In [15]:
```
# Model Evaluation
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_validate

scoring = {
    "r2": "r2",
    "sklearn MAPE": "neg_mean_absolute_percentage_error",
    "neg_root_mean_square_error": "neg_root_mean_squared_error",
```

```
    "neg_mean_squared_error": "neg_mean_squared_error"}

score_table=pd.DataFrame(cross_validate(best_model, x_train_w, y_train_w, return_tr
score_table
```

Out[15]:

	0	1	2	3	4
fit_time	0.028762	0.033563	0.031561	0.030901	0.012470
score_time	0.012424	0.005417	0.000000	0.006601	0.015561
test_r2	0.310275	0.352294	0.342595	0.062560	0.286411
train_r2	0.357499	0.347234	0.346548	0.354813	0.363308
test_sklearn MAPE	-0.097457	-0.103011	-0.097762	-0.106176	-0.102596
train_sklearn MAPE	-0.099286	-0.097894	-0.099527	-0.098261	-0.097806
test_neg_root_mean_square_error	-0.722673	-0.744341	-0.722322	-0.860427	-0.753973
train_neg_root_mean_square_error	-0.721092	-0.715690	-0.723061	-0.718781	-0.713437
test_neg_mean_squared_error	-0.522256	-0.554044	-0.521749	-0.740335	-0.568475
train_neg_mean_squared_error	-0.519973	-0.512212	-0.522818	-0.516646	-0.508992

In [16]:
```
score_table.mean(axis=1).to_frame()
```

Out[16]:

	0
fit_time	0.027451
score_time	0.008001
test_r2	0.270827
train_r2	0.353880
test_sklearn MAPE	-0.101400
train_sklearn MAPE	-0.098555
test_neg_root_mean_square_error	-0.760747
train_neg_root_mean_square_error	-0.718412
test_neg_mean_squared_error	-0.581372
train_neg_mean_squared_error	-0.516128

In [17]:
```
best_model.score(x_test_w, y_test_w)  # Get the scores using the best hyperparamete
```

Out[17]:
```
0.3198310306347695
```

Result

Model Performance:

R-squared (test): The model explains around 27% of the variance in the wine quality on the test set, indicating some predictive capability, even it's not good enough.

R-squared (train): A higher value (around 35%) on the training set suggests some fit of the model to the training data, but still can't perform well.

Error Metrics:

Negative RMSE (test): The model's root mean square error on the test set is approximately 0.76, but given the negative sign, it suggests that the model performs worse than predicting the mean value.

Negative RMSE (train): Similar to the test set, the negative RMSE on the training data is around 0.72, indicating room for improvement.

Negative MSE (test & train): Both test and training set MSEs are negative, signifying a worse performance than a model predicting the mean.

Mean Absolute Percentage Error (MAPE) & Negative MAPE (test & train): Both test and training set MAPE values are negative, indicating inaccuracies in predictions beyond simply predicting the mean.t very high.

Conclusion:

The polynomial regression model, based on the above statistical result metrics, shows some predictive ability, but it's not robust enough. The R-squared values suggest that the model explains only a moderate portion of the variance in wine quality. The negative error metrics (RMSE, MSE, and MAPE) imply that the model's performance is worse than a baseline model predicting the mean value. This indicates substantial room for improvement. Further refinement of the model may be necessary, considering feature engineering, hyperparameter tuning, or exploring alternative models to enhance predictive accuracy and better capture the complexities of wine quality determination from physicochemical properties. In summary, while the polynomial regression approach has been applied, the model's current performance suggests the need for refinement to make more accurate predictions regarding wine quality based on these physicochemical properties.

References

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems. DOI: https://doi.org/10.24432/C56S3T

Dua, Dheeru, and Casey Graff. 2019. “UCI Machine Learning Repository.” University of California, Irvine, School of Information; Computer Sciences. https://archive.ics.uci.edu/dataset/186/wine+quality.

WSET Global. (n.d.). WSET systematic approach to tasting (SAT). Retrieved from https://www.wsetglobal.com/knowledge-centre/wset-systematic-approach-to-tasting-sat/ed: Mystery Tasting. (n.d.). WSET SAT explained. Retrieved from https://www.mysterytasting.com/wset-sat-explainace.

MDS-2023-24. (2023). MDS Lecture Notes on GitHub repository. Retrieved from https://github.ubc.ca/MDS-2023-24.

Fan, J. (1996). Local Polynomial Modelling and Its Applications: From linear regression to nonlinear regression. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC. ISBN 978-0-412-98321-4.

Hoerl, A. E. (1962). Application of Ridge Analysis to Regression Problems. Chemical Engineering Progress, 58(3), 54–59.

van Wieringen, W. (2021, May 31). Lecture notes on ridge regression. arXiv:1509.09169 [stat.ME].

Tikhonov, A. N., & Arsenin, V. Y. (1977). Solution of Ill-posed Problems. Washington: Winston & Sons. ISBN 0-470-99124-0.

In [ ]: