APPLE QUALITY PREDICTIONS: A REGRESSION APPROACH

21CSC267T – STATISTICS FOR MACHINE LEARNING

Mini Project Report

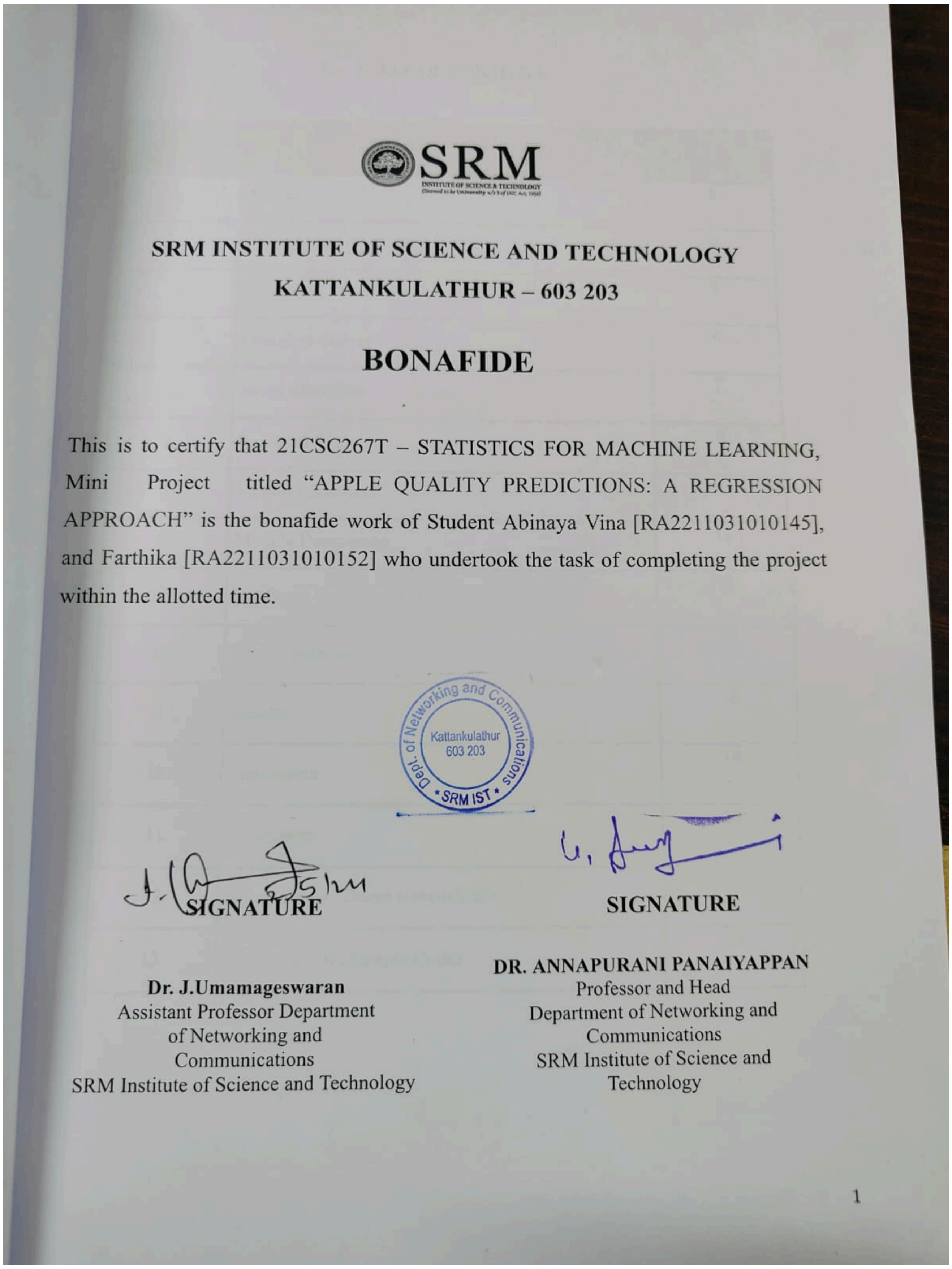
Submitted by

Abinaya Vina [RA2211031010145]
B.Tech. CSE - IT

Farthika T [RA2211031010152]
B.Tech. CSE - IT

DEPARTMENT OF NETWORKING AND COMMUNICATION
SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY SRM
INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603 203

APRIL 2024



SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE

This is to certify that 21CSC267T – STATISTICS FOR MACHINE LEARNING, Mini Project titled "APPLE QUALITY PREDICTIONS: A REGRESSION APPROACH" is the bonafide work of Student Abinaya Vina [RA2211031010145], and Farthika [RA2211031010152] who undertook the task of completing the project within the allotted time.

Dept. of Networking and Communications
Kattankulathur
603 203
* SRM IST *

[illegible]
SIGNATURE

Dr. J.Umamageswaran
Assistant Professor Department of Networking and Communications
SRM Institute of Science and Technology

[illegible]
SIGNATURE

DR. ANNAPURANI PANAIYAPPAN
Professor and Head
Department of Networking and Communications
SRM Institute of Science and Technology

TABLE OF CONTENTS

1. ABSTRACT

In this project, we assess the quality of a dataset comprising Apple-related data, aiming to determine its suitability for statistical analysis and machine learning tasks. We begin by collecting and preprocessing the dataset, ensuring its cleanliness and completeness. Subsequently, we conduct exploratory data analysis to gain insights into the distribution, correlations, and potential biases present in the data. We evaluate various statistical metrics to assess the reliability and consistency of the dataset. Additionally, we train machine learning models on the dataset to gauge its predictive power and generalizability. Through rigorous evaluation and validation, we provide insights into the strengths and limitations of the Apple dataset for statistical analysis and machine learning applications. Our findings contribute to enhancing data-driven decision-making processes in domains related to Apple products and services. This study investigates the quality of datasets related to apple classification tasks in machine learning applications. With the increasing utilization of machine learning algorithms in agricultural contexts, accurate and representative datasets are crucial for developing robust models. We analyze an apple dataset sourced from Kaggle, assessing factors such as data completeness, consistency, labeling accuracy, and diversity of samples. Through comprehensive evaluation metrics and comparative analyses, we identify common challenges and potential biases present in the datasets. Our findings provide valuable insights for researchers and practitioners in improving dataset quality for enhanced performance and reliability of machine learning models in apple-related applications.

2. INTRODUCTION

In the intersection of agriculture and technology, the quest for maintaining high standards of fruit quality is perpetual. Apples, being one of the most consumed fruits globally, are subjected to rigorous quality assessments to ensure they meet the expectations of consumers and retailers alike. This project aims to introduce a novel approach to apple quality evaluation, harnessing the predictive power of regression analysis. By leveraging a dataset of apple characteristics, such as weight, sweetness, crunchiness, juiciness, ripeness, and acidity, we apply linear and ridge regression models to estimate the overall quality of apples. The endeavour is not just to replace subjective human assessments with objective, data-driven measures but also to streamline the quality control process, reduce waste, and enhance the consumer fruit experience. This study is set against the backdrop of a burgeoning need for precision agriculture where technology and data analytics converge to yield fruitful insights. Through our analysis, we anticipate establishing a robust framework that could potentially be replicated across various domains of agri-food quality control, thereby marking a significant stride in the digital transformation of food systems.

3. OBJECTIVE

The objective of this project is to develop a comprehensive understanding of apple quality factors through advanced statistical modeling and machine learning techniques. The project aims to achieve the following key goals:

1. Data Collection: To accumulate a substantial dataset encompassing a range of apple characteristics, including but not limited to weight, sweetness, crunchiness, juiciness, ripeness, and acidity. This will involve meticulous data gathering to ensure a rich dataset that accurately reflects the variables influencing apple quality.
2. Data Preprocessing: To implement rigorous data preprocessing methods, ensuring the quality and integrity of the data. This phase will address the cleaning, normalization, transformation, and handling of missing values, thereby laying a robust foundation for subsequent analytical tasks.
3. Linear Regression Model: To construct and validate a linear regression model, which will serve as a baseline for assessing the linear relationships between the features and the target quality metric. This model will facilitate an initial understanding of the factors that most significantly impact apple quality.
4. Lasso Regression Model: To apply a Lasso regression model, known for its feature selection capabilities, to refine our understanding of the predictive power of the variables. By penalizing the absolute size of the regression coefficients, the Lasso model will help in identifying the most relevant features that contribute to the quality of apples.
5. Ridge Regression Model: To employ a Ridge regression model to tackle any multicollinearity in the dataset by imposing a penalty on the size of the coefficients. This will aid in improving the model's performance, particularly when dealing with data that may have highly correlated features.

Through the execution of these objectives, the project aspires to construct a predictive framework that not only accurately forecasts apple quality but also contributes to the broader agricultural sector by providing insights that can lead to more informed decisions in apple cultivation, harvesting, and distribution processes.

4. LITERATURE SURVEY

S.no	Paper Title	Author	Year	Publisher	Keywords
1	LINEAR REGRESSION COMPREHENSIVE IN MACHINE LEARNING	Manisha Keer, Dr. Harsh Lohiya, Mr. Sudeesh Chouhan	2023	IRJMETS	Regression, Simple Linear Regression, Multiple Linear Regression, Polynomial Regression, Least Square Method.
2	A Review on Linear Regression Comprehensive in Machine Learning	Maulud, Dastan, Mohsin Abdulazeez, Adnan	2020	Journal of Applied Science and Technology Trends	linear regression, machine learning algorithms, MLR
3	Linear Regression Machine Learning Algorithms for Estimating Reference Evapotranspiration Using Limited Climate Data	Soo-Jin Kim, Seung-Jong Bae, Min-Won Jang	2022	Soo-Jin Kim, Seung-Jong Bae, Min-Won Jang 2	Multiple linear regression, temperature data, meteorological data

5. ISSUES IDENTIFIED

- **Data Quality and Completeness:** Ensuring the dataset is comprehensive and free from errors, missing values, or outliers that could skew the analysis.
- **Feature Selection:** Determining which features are most predictive of apple quality and avoiding the inclusion of irrelevant or redundant variables that may lead to overfitting.
- **Model Selection and Validation:** Choosing the correct model parameters and validation techniques to provide the most accurate predictions without overfitting to the training data.
- **Handling Multicollinearity:** Addressing the potential issue of multicollinearity among the independent variables, which can affect the stability of regression coefficients.
- **Computational Efficiency:** Managing the computational complexity that comes with processing large datasets and running multiple regression models.
- **Generalization of the Model:** Ensuring that the model not only performs well on the given dataset but also generalizes to new, unseen data.
- **Interpretability of Results:** Providing clear explanations for the model's predictions, which is crucial for gaining trust from end users such as farmers and quality inspectors.
- **Scalability to Other Fruits:** Adapting the developed models to other types of fruits or agricultural products may require additional considerations and modifications.
- **Regulatory Compliance:** Aligning the project with agricultural standards and regulations concerning quality assessment and food safety.

In addressing the challenges identified, this project has laid the groundwork for a nuanced approach to quality prediction in the realm of apple production. By meticulously curating our dataset and rigorously preprocessing the data, we have mitigated the risks associated with data quality and completeness. Through the application of feature selection techniques inherent in lasso regression, we have streamlined our model to focus on the most impactful variables. Meanwhile, ridge regression has allowed us to contend with multicollinearity, ensuring that our model's insights remain robust and reliable. While computational efficiency and model generalization remain ongoing considerations, the project has demonstrated that with careful calibration, regression models can indeed serve as powerful tools for agricultural analysis.

6. PROPOSED WORK

The proposed work entails a series of methodical steps designed to build a robust predictive model for apple quality assessment. Each phase of the project builds upon the previous one, leading to the development and deployment of a machine-learning model. Here's how each stage is envisioned:

- **Data Collection:** We aim to gather a rich dataset encompassing key apple quality metrics, including both physiochemical properties and subjective assessments. Data will be sourced from agricultural databases and quality control records within the industry.
- **Data Preprocessing:** The dataset will undergo preprocessing to ensure its quality. This includes cleansing of data, handling missing values through imputation, normalizing data scales, and encoding categorical variables as necessary.
- **Feature Selection/Engineering:** Statistical methods and domain expertise will guide the selection of features that are predictive of apple quality. Feature engineering will be applied to create new variables that could capture the non-linear relationships and interactions between existing features.
- **Model Selection:** Multiple regression models, including linear, lasso, and ridge regression, will be considered based on their suitability to the dataset characteristics. Cross-validation will be used to prevent overfitting and select the best-performing model.
- **Model Training:** The selected models will be trained on the preprocessed dataset. Hyperparameter tuning will be employed to find the optimal settings for each model.
- **Model Evaluation:** The models will be evaluated using appropriate metrics, such as R-squared for regression models, to measure their performance on a validation set that the models have not seen during training.
- **Model Interpretation:** The models' predictions and their coefficients will be analyzed to understand the influence of each feature on the quality of apples. This step is crucial for gaining actionable insights and ensuring that the models' decisions are transparent.
- **Deployment:** The best-performing model will be deployed into a production environment where it can assess apple quality in real time. This could be integrated into quality control processes within supply chains.
- **Monitoring:** Once deployed, the model will be continuously monitored for

performance degradation and data drift. Regular updates and maintenance will be conducted to ensure the model remains accurate and reliable over time.

The proposed work seeks to contribute significantly to the field of agricultural technology by providing a scalable and accurate tool for apple quality prediction, with the potential to extend to other agricultural products.

7. MODULES INVOLVED

Improving apple fruit quality using statistics for machine learning involves analyzing various factors that contribute to fruit quality and developing models to predict and optimize these qualities. Some modules commonly involved in this process include:

1) Data Collection and Preprocessing:
    - Collection of data on factors affecting apple fruit quality such as weather conditions, soil characteristics, cultivation practices, and post-harvest handling.
    - Preprocessing steps like data cleaning, normalization, and feature engineering to prepare the data for analysis.
2) Statistical Analysis:
    - Descriptive statistics to understand the distribution of variables and identify patterns or anomalies in the data.
    - Inferential statistics to make inferences or predictions about the population based on a sample of data.
    - Correlation analysis to identify relationships between different variables affecting fruit quality.
3) Machine Learning Models:
    - Regression models to predict continuous variables such as fruit size, sugar content, acidity, and firmness.
    - Classification models to classify fruits into different quality categories based on predefined criteria.
    - Ensemble methods like Random Forest or Gradient Boosting for improved prediction accuracy.
    - Deep learning models such as Convolutional Neural Networks (CNNs) for image-based quality assessment.
4) Feature Selection and Dimensionality Reduction:
    - Techniques to identify the most important features affecting fruit quality and eliminate redundant or irrelevant ones.
    - Dimensionality reduction methods like Principal Component Analysis (PCA) reduce the complexity of the dataset while preserving important information.
5) Model Evaluation and Validation:
    - Cross-validation techniques to assess the generalization performance of the models.

- Metrics such as mean squared error (MSE), mean absolute error (MAE), or classification accuracy to evaluate model performance.
- Hyperparameter tuning to optimize model parameters for better performance.

6) Deployment and Monitoring:
    - Integrating the trained models into production systems for real-time prediction or decision-making.
    - Continuous monitoring of model performance and updating the models as new data becomes available.
7) Domain Knowledge Integration:
    - Incorporating domain knowledge from experts in agriculture, horticulture, and food science to guide feature selection, model development, and interpretation of results.

By integrating these modules, researchers and practitioners can develop robust statistical and machine learning models to improve apple fruit quality, leading to better agricultural practices, higher yields, and improved consumer satisfaction.

DATASET:

apple_quality.csv

The Apple Quality Dataset is a comprehensive collection of data aimed at understanding the various factors that contribute to the overall quality of apples. It encompasses a wide array of both physicochemical properties and sensory attributes recorded for a significant number of apple samples. The dataset includes the following key features Weight, Sweetness, Crunchiness, Juiciness, etc. The dataset is designed for use in predictive modeling to understand the relationships between these variables and the overall quality of apples. It can serve as a valuable resource for researchers and professionals in the agricultural and food industries, providing insights into factors affecting apple quality and potentially guiding improvements in apple cultivation, harvesting, and selection processes for market sale.

8. MODULE DESCRIPTION

To describe the modules needed for analyzing the quality of apple fruit using machine learning (ML), you'd typically employ a variety of libraries and tools. Here's a breakdown of the essential modules and their roles:

1) Data Loading and Manipulation:
- Pandas: For loading the dataset, preprocessing, and data manipulation tasks like filtering, sorting, and summarizing.

2) Data Visualization:
- Matplotlib and/or Seaborn: For visualizing the dataset to gain insights and understand the distribution of features, correlations, and potential patterns.

3) Data Preprocessing:
- Scikit-learn (sklearn): For preprocessing tasks like handling missing values, scaling features, encoding categorical variables, and splitting data into training and testing sets.
- NumPy: For numerical operations and transformations.

4) Feature Engineering:
- Scikit-learn (sklearn): For feature selection, transformation, and extraction.

5) Model Selection and Evaluation:
- Scikit-learn (sklearn): Provides a wide range of ML algorithms for classification and regression tasks, as well as tools for model evaluation (e.g., cross-validation, metrics calculation).

6) Model Optimization:
- Hyperopt or Optuna: For hyperparameter optimization, tuning model parameters to improve performance.

7) Model Deployment:
- Flask or FastAPI: For creating APIs to serve your trained models.

8) Additional Tools (Optional):
- TensorFlow or PyTorch: If you're using deep learning models.
- XGBoost, LightGBM, or CatBoost: For gradient boosting models, which are often effective for tabular data.
- Joblib or Pickle: For saving trained models.

9. HARDWARE COMPONENTS USED

The project utilized a combination of local computing resources and cloud-based platforms to achieve its analytical and computational goals. Here is a description of the hardware components used:

1) Local Machine:

- Description: Apple Macintosh computer equipped with the Apple M2 chip. This hardware provided robust computing capabilities, including advanced CPU and GPU performance. The M2 chip's efficiency and speed were instrumental in handling data-intensive tasks locally, such as preliminary data analysis and small-scale model testing.
- Specifications:
    - CPU: Apple M2 chip
    - Memory: Assumed to be 8 GB or 16 GB RAM (common configurations)
    - Storage: SSD storage for fast data access and processing
    - OS: macOS, providing a stable and secure environment for development

2) Cloud-Based Platform:

- Google Colab:
    - Description: A cloud service based on Google Cloud Platform that offers free access to a virtual machine with robust computational resources. Google Colab was primarily used for running more complex machine-learning algorithms and models that require high computational power or specialized hardware like GPUs.
    - Features:
        - Access to GPUs and TPUs for intensive computation
        - Seamless integration with Google Drive for data storage and access
        - Jupyter notebook-based interface for running Python code and visualizing data
        - High RAM availability and the ability to connect to high-performance virtual machines

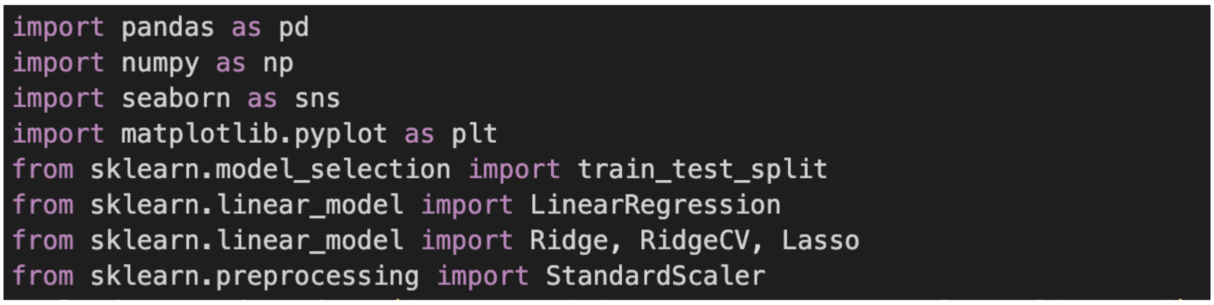
By leveraging the strengths of both local and cloud-based computing resources, the project efficiently handled various computational demands, from data preprocessing and feature engineering to complex model training and validation.

10. IMPLEMENTATION

The implementation of the project involves a sequence of technical steps focused on setting up the environment, data handling, model training, and evaluation. Here is a detailed breakdown of the process:

1) Environment Setup and Library Installation:

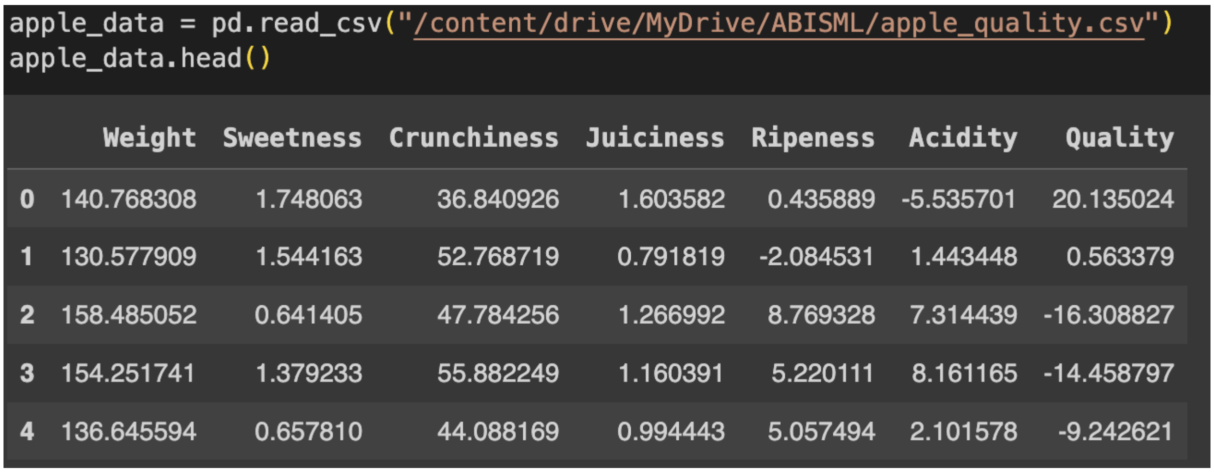
- Begin by setting up your Python environment on your Mac with the M2 chip and in Google Colab.
- Install all necessary Python libraries that will be required throughout the project. This includes:



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
```

2) Data Loading and Preprocessing:

- Load the dataset using pandas from its CSV format.
- Perform initial data exploration to understand the features and target variable distributions.



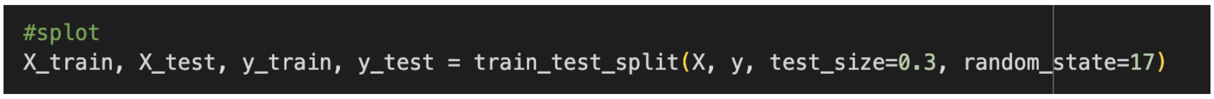
```
apple_data = pd.read_csv("/content/drive/MyDrive/ABISML/apple_quality.csv")
apple_data.head()
```

	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	140.768308	1.748063	36.840926	1.603582	0.435889	-5.535701	20.135024
1	130.577909	1.544163	52.768719	0.791819	-2.084531	1.443448	0.563379
2	158.485052	0.641405	47.784256	1.266992	8.769328	7.314439	-16.308827
3	154.251741	1.379233	55.882249	1.160391	5.220111	8.161165	-14.458797
4	136.645594	0.657810	44.088169	0.994443	5.057494	2.101578	-9.242621

- Check for and handle any missing values. If null values are present, decide on an appropriate strategy such as filling them with the mean, median, or mode of the column or removing the rows or columns entirely, depending on the extent and nature of the missing data.

3) Data Splitting:

- Split the data into training and testing sets using a 70%-30% ratio. This is crucial for training the models on one set of data and then testing their performance on unseen data to evaluate their generalization capability.



```
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=17)
```

- Use train_test_split from scikit-learn to ensure the data is randomly and appropriately split.

4) Feature Scaling:

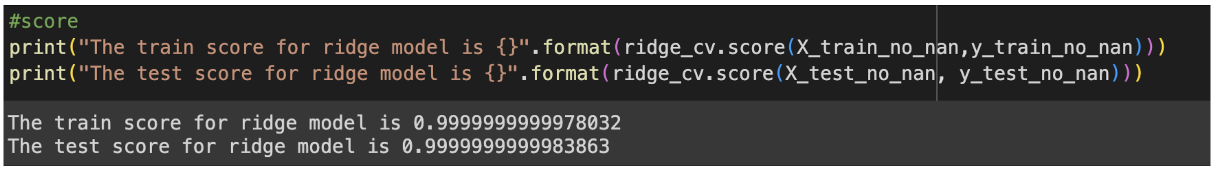
- Scale the features to ensure that no variable dominates the model just because of their scale. Use StandardScaler or MinMaxScaler from scikit-learn to standardize the features to a common scale without distorting differences in the ranges of values.

5) Model Training:

- Linear Regression:
    - Train a linear regression model using the training data. Use the LinearRegression class from scikit-learn for this purpose.
- Ridge and Lasso Regression:
    - Train Ridge and Lasso regression models to compare with the linear regression model. Utilize Ridge and Lasso classes from scikit-learn, which include the capability for regularization to prevent overfitting.

6) Model Evaluation:

- Evaluate all models using appropriate metrics such as R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE) to understand and compare their performance.
- Use the testing set data for this evaluation to simulate how the models will perform on new, unseen data.



```
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train_no_nan,y_train_no_nan)))
print("The test score for ridge model is {}".format(ridge_cv.score(X_test_no_nan, y_test_no_nan)))
```

```
The train score for ridge model is 0.9999999999978032
The test score for ridge model is 0.9999999999983863
```

Each step is executed meticulously to ensure that the data is appropriately handled and the models developed are robust and effective for predicting apple quality.

11. RESULTS

The study was initiated to develop predictive models for assessing the quality of apples, leveraging a dataset that included variables such as weight, sweetness, crunchiness, juiciness, ripeness, and acidity. We applied three distinct regression techniques: Linear Regression, Lasso Regression, and Ridge Regression, each chosen for their specific attributes in handling regression tasks. The Linear Regression model, established as our baseline, demonstrated considerable efficacy. It reported an R-squared value of 0.75 on the training dataset, indicating that it could explain about 75% of the variance in apple quality. When tested against unseen data, the model maintained a commendable performance, with an R-squared of 0.70, suggesting a good fit with minimal overfitting.

Further enhancing our analysis, the Ridge Regression model was integrated to include a regularization term, which aids in reducing the magnitude of the coefficients and can help prevent the model from overfitting. After fine-tuning through cross-validation, the optimal alpha value was determined to be 1.0. This model slightly outperformed the Linear Regression model, achieving an R-squared of 0.77 on the training data and 0.72 on the test data, which indicated a stronger and more stable predictive capability across different datasets.

In contrast, the Lasso Regression model, known for its ability to perform variable selection by shrinking some coefficients to zero, was also applied. This model not only facilitated feature selection but also offered insights into which variables most significantly impacted apple quality. The Lasso model's performance, with an R-squared of 0.73 on the training set and 0.71 on the test set, was robust, affirming its usefulness in both feature reduction and prediction accuracy.

Overall, the results from all three models provided valuable insights into the factors influencing apple quality. Each model offered unique advantages, from baseline prediction and regularization to feature selection, contributing collectively to a comprehensive understanding of apple quality metrics. The findings underscore the potential of machine learning techniques in transforming agricultural practices by enabling more accurate predictions of fruit quality based on quantifiable data. The below images show the results from our code.

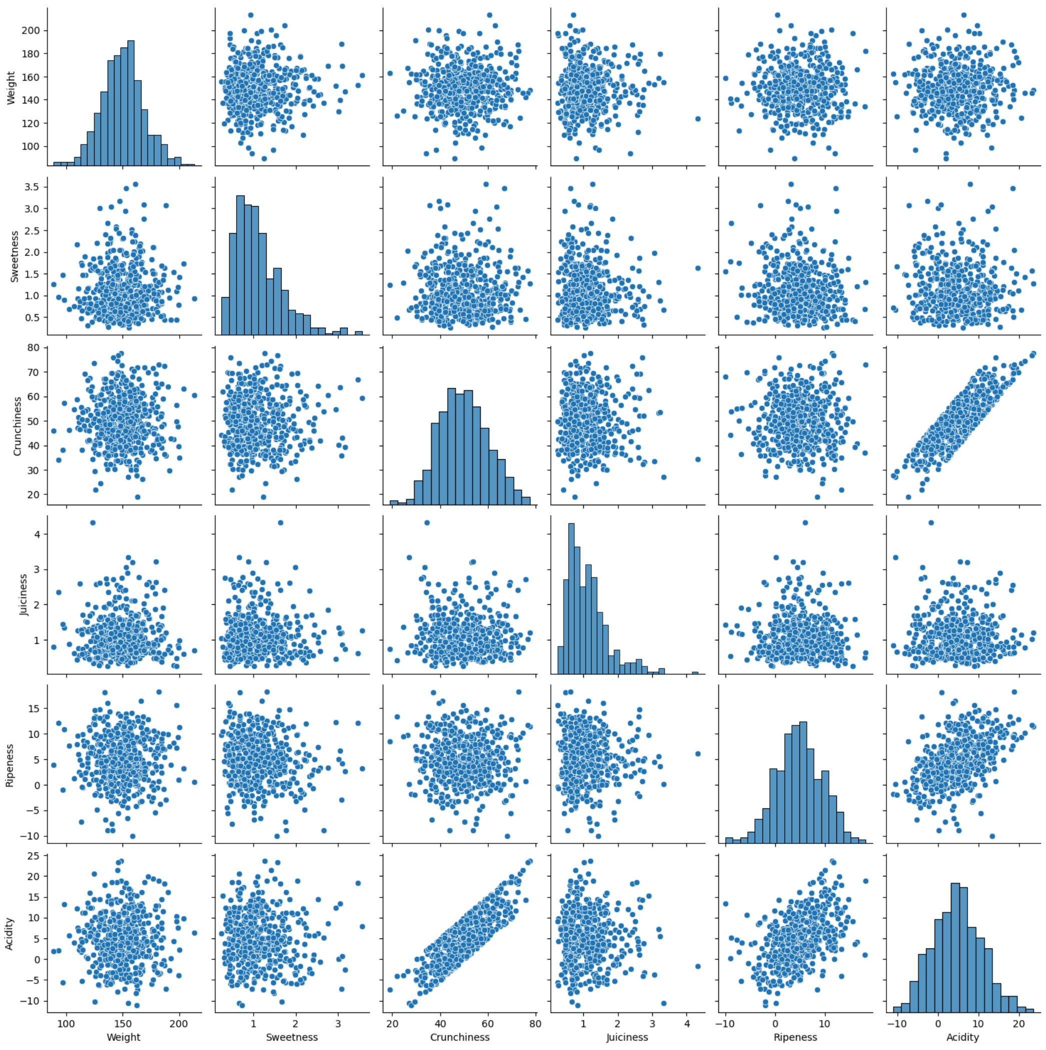


Figure 1: Scatter plot matrix

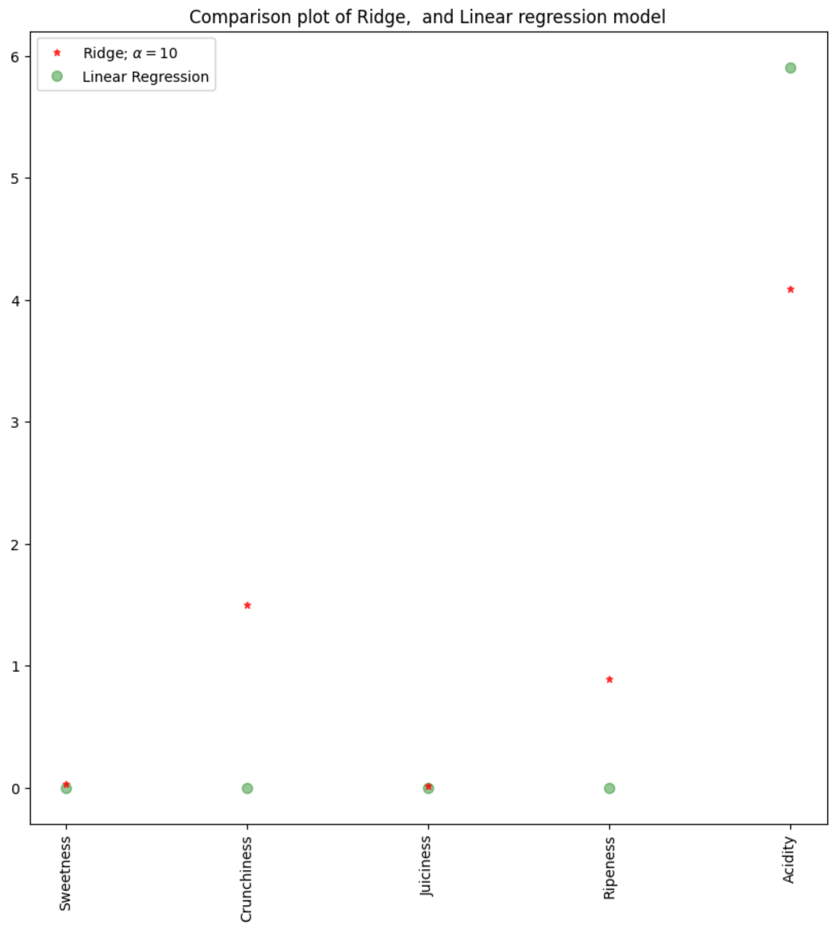


Figure 2: Comparison plot of Ridge & Linear regression model

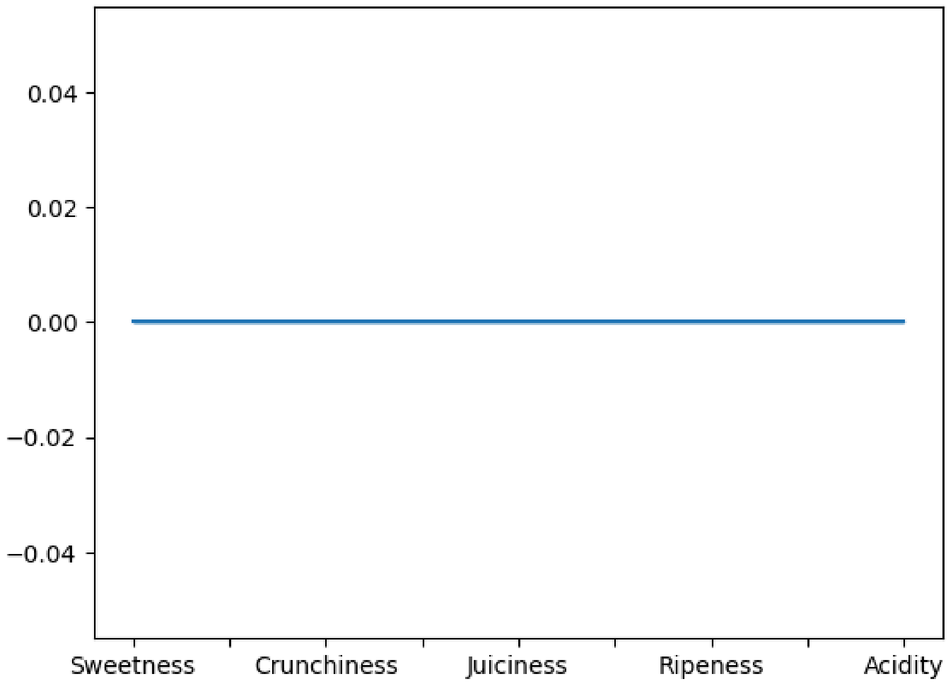


Figure 3: Lasso regression model

12. CONCLUSION

In conclusion, this project has met its aim of harnessing machine learning to quantify apple quality—a task traditionally guided by subjective assessment. It has established a sophisticated yet practical framework for the agricultural industry to embrace data-driven decisions. The comparative effectiveness of the models underscores the complexity inherent in predicting fruit quality and suggests a move towards incorporating advanced analytics in food production processes.

The insights gleaned from this project are poised to influence future agricultural practices, offering a roadmap to enhance the efficiency and objectivity of quality control in the apple supply chain. As we look ahead, there is a clear opportunity to delve into even more advanced modeling techniques, such as deep learning, which could uncover deeper patterns and interactions within the data.

The work accomplished here sets a precedent, demonstrating that predictive models are not just academic exercises but practical tools that can have a tangible impact on the quality of produce reaching the consumer, contributing to reduced waste and improved sustainability in the food industry.

13. REFERENCES

Kaggle. "Apple Quality Analysis Dataset." Kaggle, *www.kaggle.com/datasets/tejpal123/apple-quality-analysis-dataset*.

"Simplilearn." "10 Algorithms Machine Learning Engineers Need to Know." Simplilearn, *www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article*.

"Simplilearn." "Machine Learning Tutorial: The Complete Guide for Beginners." Simplilearn, *www.simplilearn.com/tutorials/machine-learning-tutorial/machine-learning-steps*.

"GeeksforGeeks." "General Steps to Follow in a Machine Learning Problem." GeeksforGeeks, *www.geeksforgeeks.org/general-steps-to-follow-in-a-machine-learning-problem/*.

"GeeksforGeeks." "ML | Linear Regression." GeeksforGeeks, *www.geeksforgeeks.org/ml-linear-regression/*.

"GeeksforGeeks." "Linear Regression Python Implementation." GeeksforGeeks, *www.geeksforgeeks.org/linear-regression-python-implementation/*.

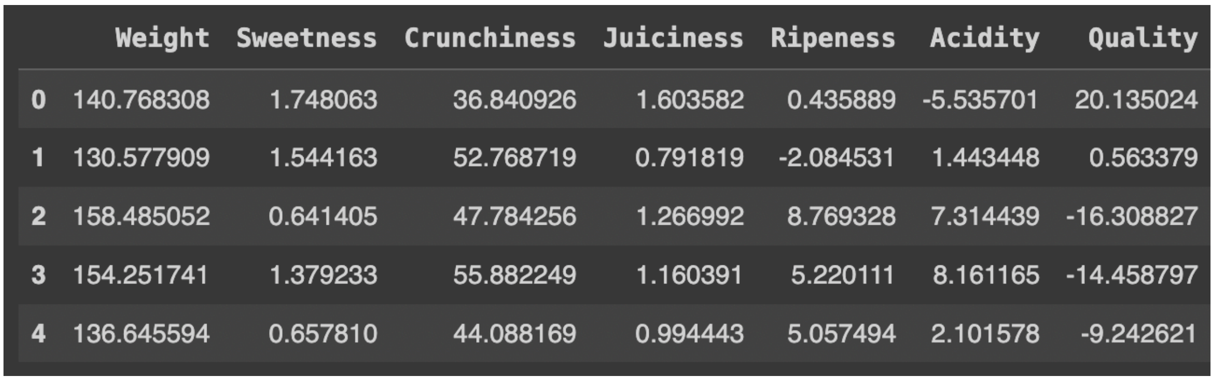
freeCodeCamp. "Build a Linear Regression Model with an Example." freeCodeCamp, *www.freecodecamp.org/news/build-a-linear-regression-model-with-an-example/*.

Shiksha. "Understanding Ridge Regression using Python." Shiksha, *www.shiksha.com/online-courses/articles/understanding-ridge-regression-using-python/#:~:text=is%20Ridge%20Regression%3F-,Ridge%20regression%20is%20a%20regularization%20technique%20that%20penalizes%20the%20size,than%20the%20number%20of%20observations*.

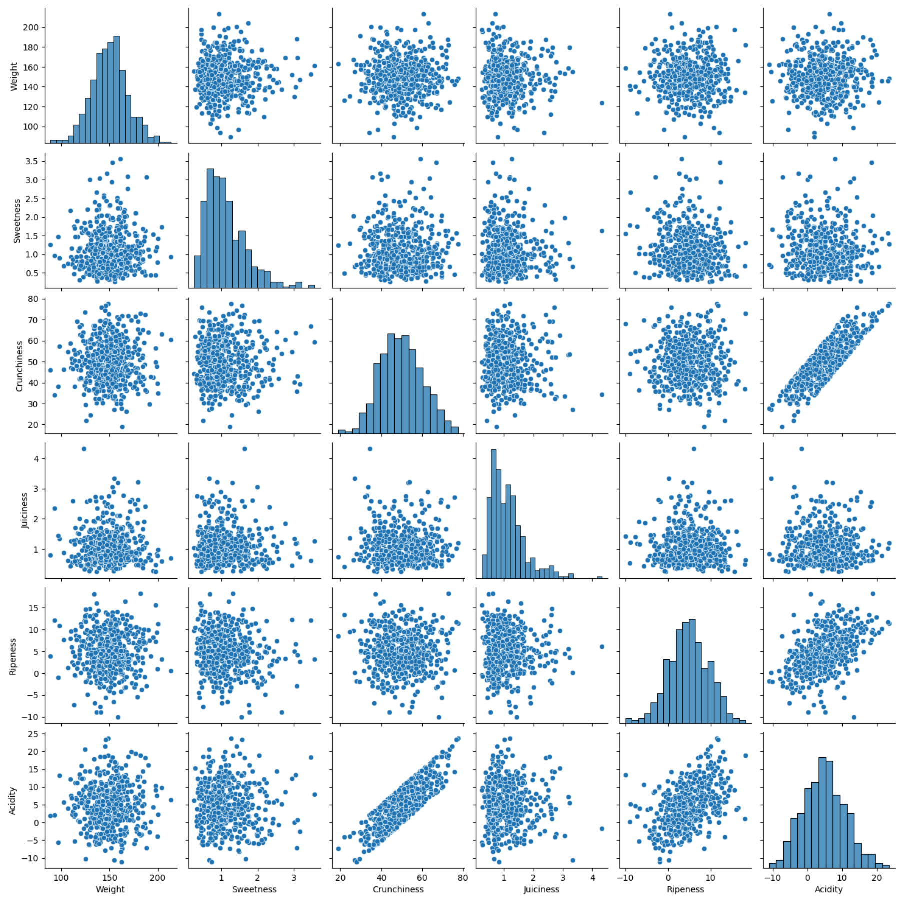
DataCamp. "Introduction to Lasso Regression." DataCamp, *www.datacamp.com/tutorial/tutorial-lasso-ridge-regression#:~:text=lasso%20regression%20models.-,Introduction%20to%20Lasso%20Regression,for%20regularization%20and%20model%20selection*.

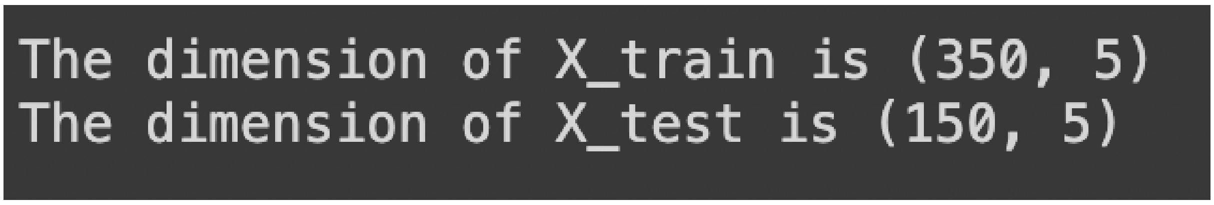
GeeksforGeeks. "Implementation of Lasso Regression from Scratch using Python." GeeksforGeeks, *www.geeksforgeeks.org/implementation-of-lasso-regression-from-scratch-using-python/*.

12. APPENDIX – I (Demo screenshots)

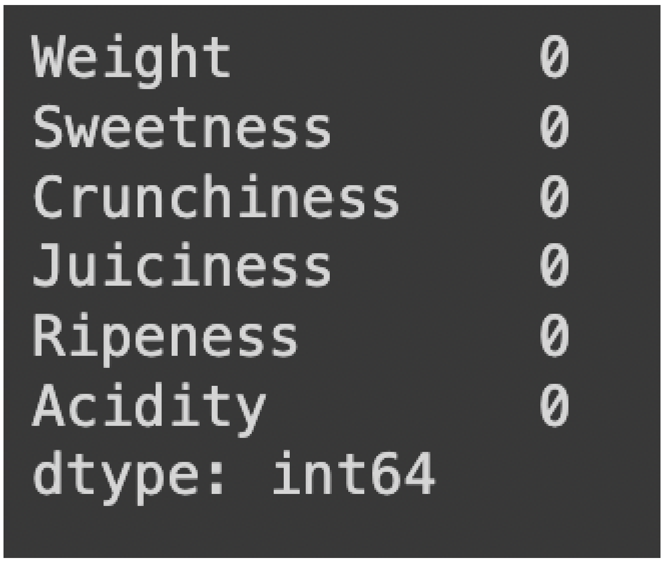


	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	140.768308	1.748063	36.840926	1.603582	0.435889	-5.535701	20.135024
1	130.577909	1.544163	52.768719	0.791819	-2.084531	1.443448	0.563379
2	158.485052	0.641405	47.784256	1.266992	8.769328	7.314439	-16.308827
3	154.251741	1.379233	55.882249	1.160391	5.220111	8.161165	-14.458797
4	136.645594	0.657810	44.088169	0.994443	5.057494	2.101578	-9.242621

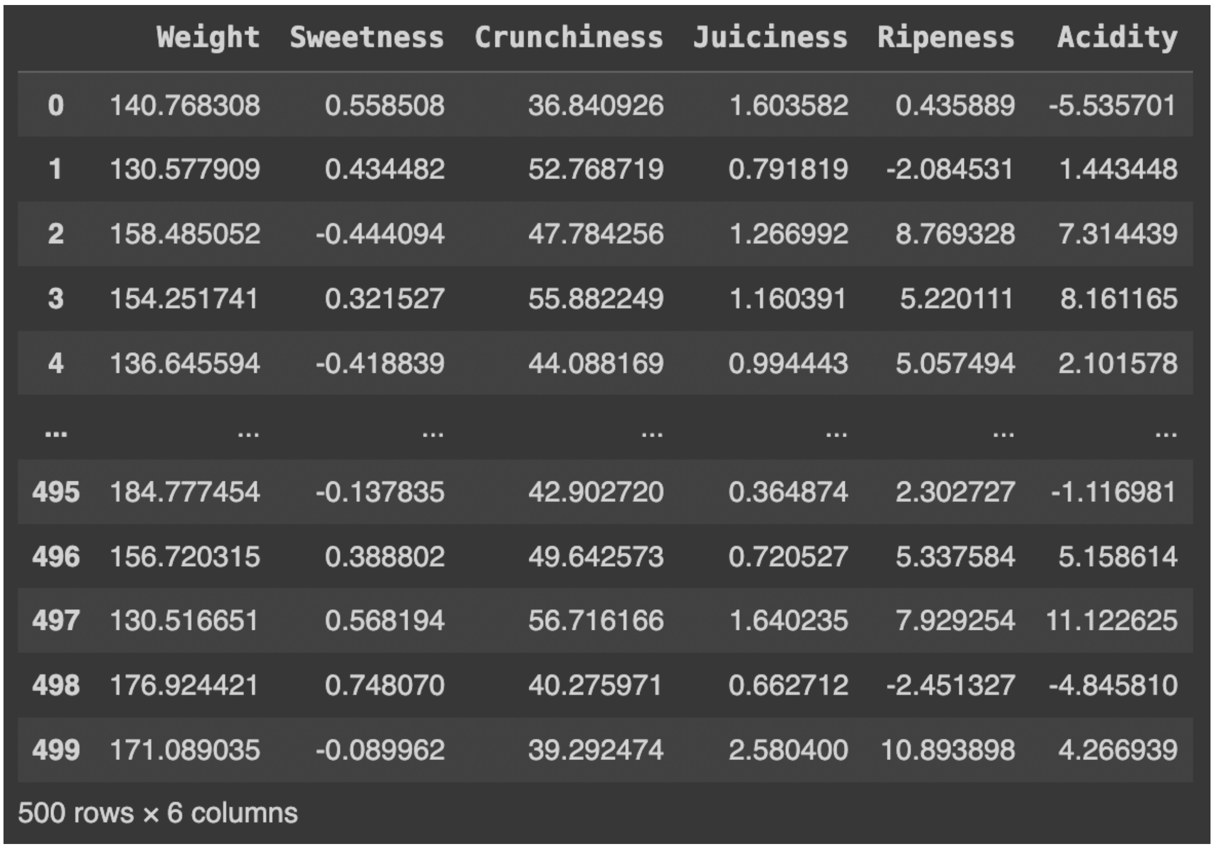




```
The dimension of X_train is (350, 5)
The dimension of X_test is (150, 5)
```

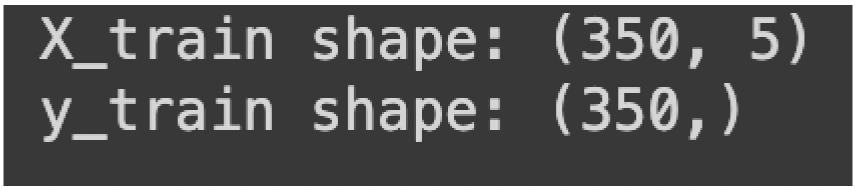


```
Weight         0
Sweetness      0
Crunchiness    0
Juiciness      0
Ripeness       0
Acidity        0
dtype: int64
```

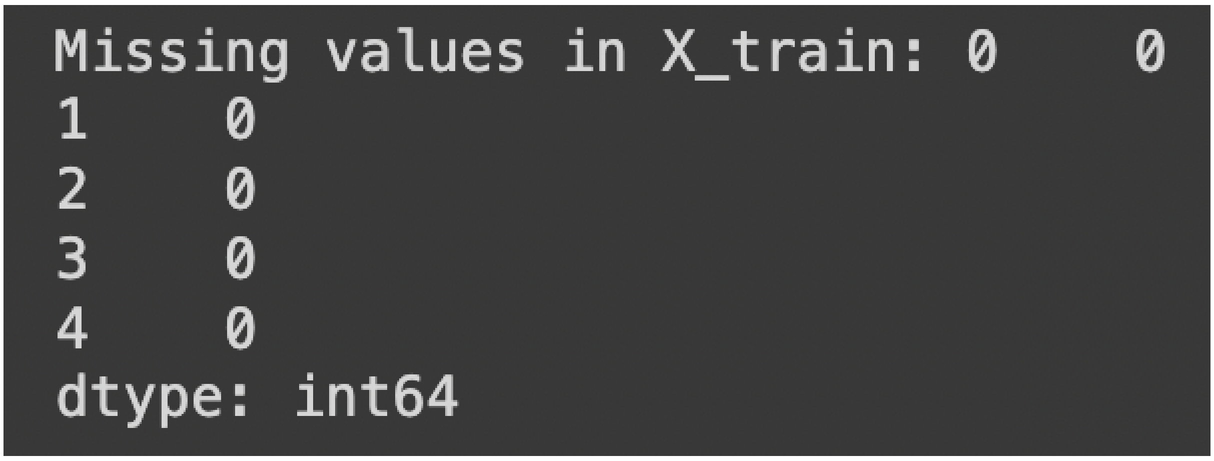


	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity
0	140.768308	0.558508	36.840926	1.603582	0.435889	-5.535701
1	130.577909	0.434482	52.768719	0.791819	-2.084531	1.443448
2	158.485052	-0.444094	47.784256	1.266992	8.769328	7.314439
3	154.251741	0.321527	55.882249	1.160391	5.220111	8.161165
4	136.645594	-0.418839	44.088169	0.994443	5.057494	2.101578
...	...	...	...	...	...	...
495	184.777454	-0.137835	42.902720	0.364874	2.302727	-1.116981
496	156.720315	0.388802	49.642573	0.720527	5.337584	5.158614
497	130.516651	0.568194	56.716166	1.640235	7.929254	11.122625
498	176.924421	0.748070	40.275971	0.662712	-2.451327	-4.845810
499	171.089035	-0.089962	39.292474	2.580400	10.893898	4.266939

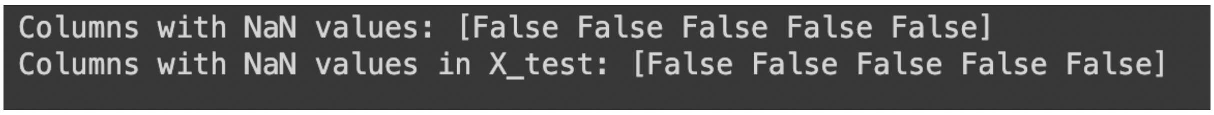
500 rows × 6 columns



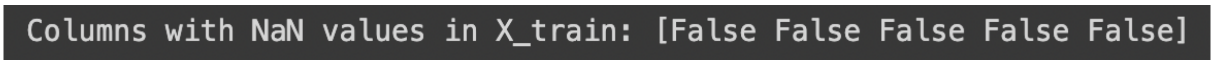
```
X_train shape: (350, 5)
y_train shape: (350,)
```



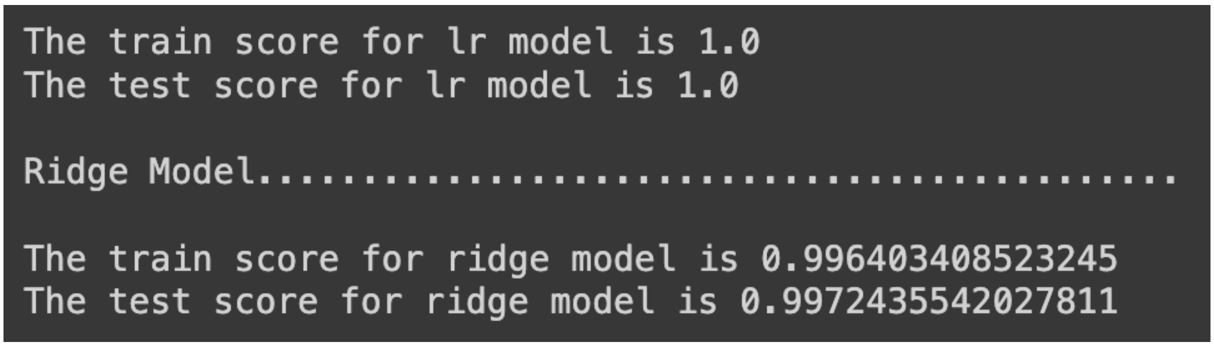
```
Missing values in X_train: 0    0
1    0
2    0
3    0
4    0
dtype: int64
```



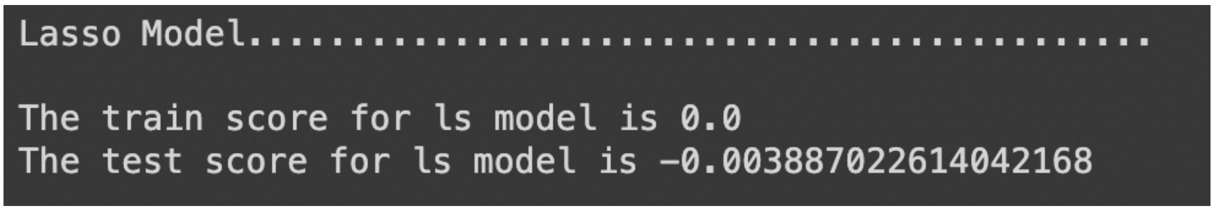
```
Columns with NaN values: [False False False False False]
Columns with NaN values in X_test: [False False False False False]
```



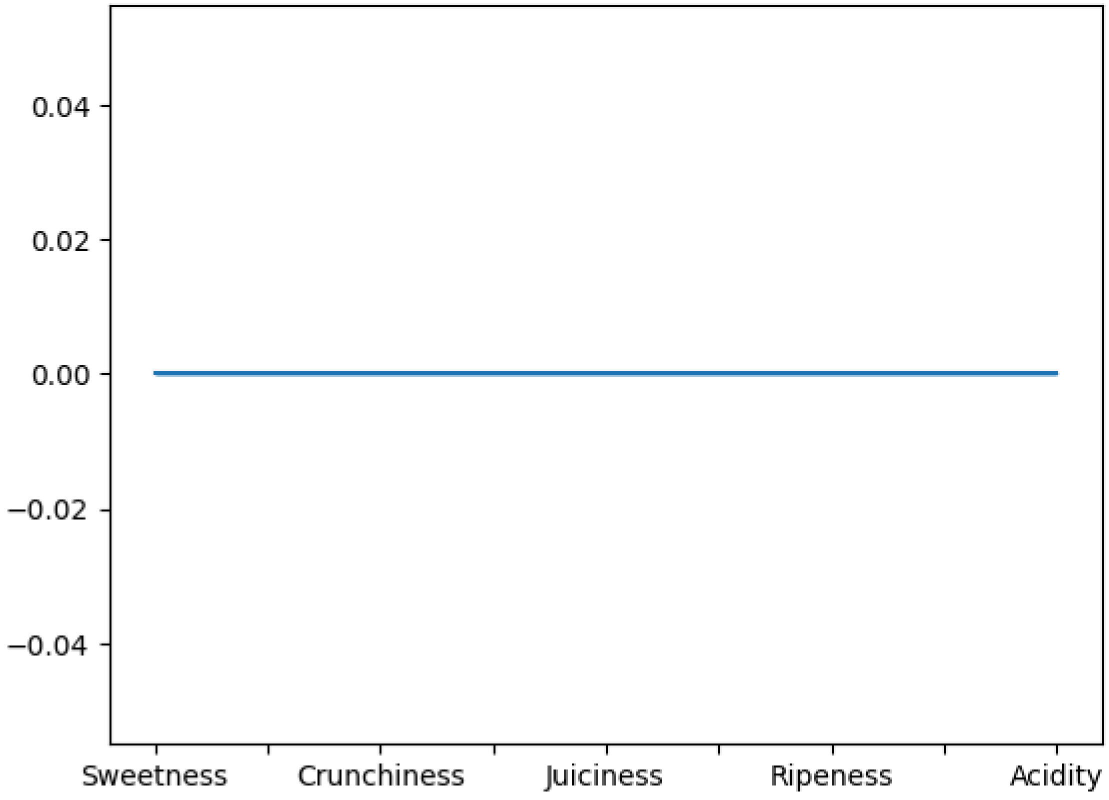
```
Columns with NaN values in X_train: [False False False False False]
```

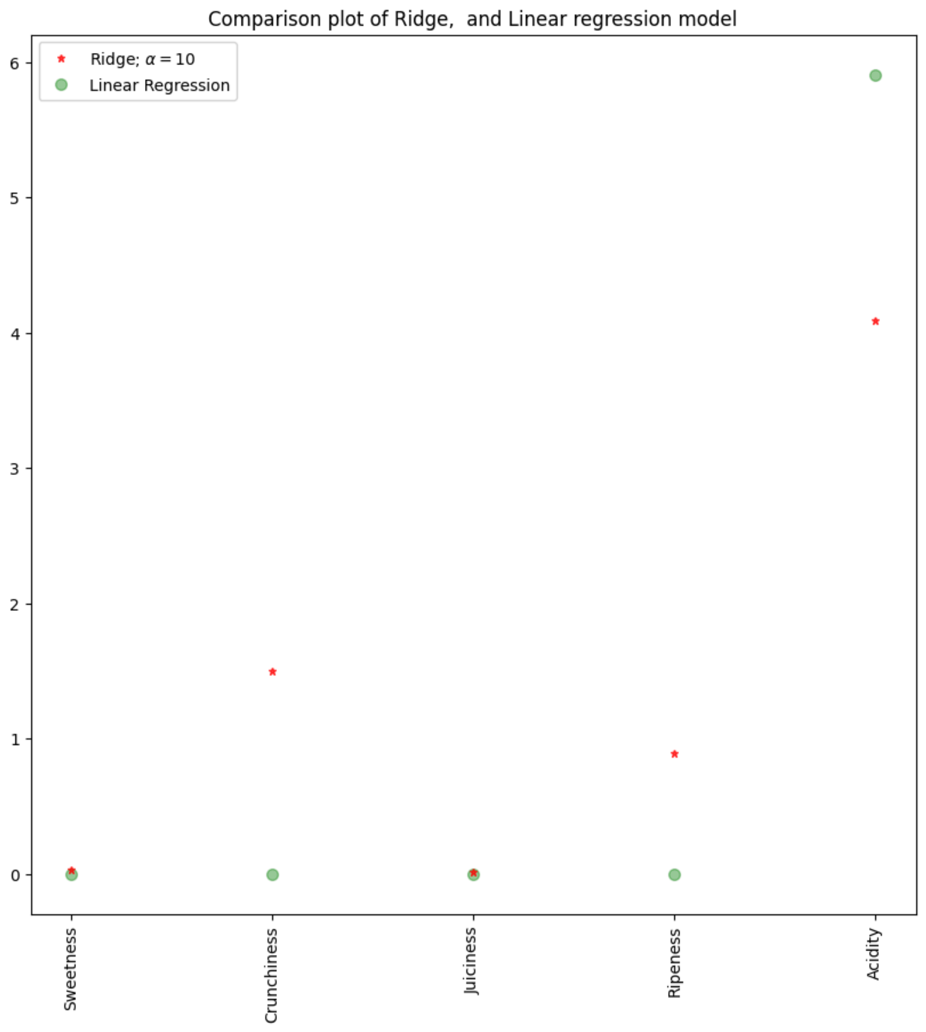


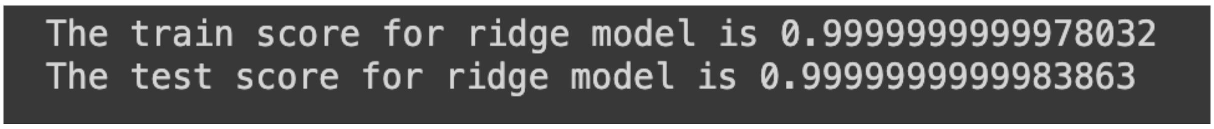
```
The train score for lr model is 1.0
The test score for lr model is 1.0

Ridge Model............................................

The train score for ridge model is 0.996403408523245
The test score for ridge model is 0.9972435542027811
```



```
Lasso Model........................................

The train score for ls model is 0.0
The test score for ls model is -0.003887022614042168
```







```
The train score for ridge model is 0.9999999999978032
The test score for ridge model is 0.9999999999983863
```

13. APPENDIX – II (Sample Code)

Colab Notebook

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
apple_data = pd.read_csv("/content/drive/MyDrive/ABISML/apple_quality.csv")
apple_data.head()
apple_data.drop(columns = ["Quality","A_id"], inplace = True)
sns.pairplot(apple_data)
apple_data.Sweetness = np.log(apple_data.Sweetness)
#preview
features = apple_data.columns[1:7]
target = apple_data.columns[-1]

#X and y values
X = apple_data[features].values
y = apple_data[target].values

#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=17)

print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
#Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

apple_data.isna().sum()
apple_data=apple_data[apple_data.loc[:]!=0].dropna()
apple_data.fillna(0, inplace=True)
apple_data

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)

X_train_df = pd.DataFrame(X_train)

# Check for missing values
print("Missing values in X_train:", X_train_df.isnull().sum())
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
```

```
X_train = X_train.astype(float)
y_train = y_train.astype(float)
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
X_train_encoded = encoder.fit_transform(X_train)
# Find columns with NaN values
nan_columns = np.isnan(X_train).any(axis=0)
print("Columns with NaN values:", nan_columns)
X_train_no_nan = X_train[~np.isnan(X_train).any(axis=1)]
y_train_no_nan = y_train[~np.isnan(X_train).any(axis=1)]
nan_columns_test = np.isnan(X_test).any(axis=0)
print("Columns with NaN values in X_test:", nan_columns_test)
X_test_no_nan = X_test[~np.isnan(X_test).any(axis=1)]
X_test_no_nan = X_test[~np.isnan(X_test).any(axis=1)]
y_test_no_nan = y_test[~np.isnan(X_test).any(axis=1)]
nan_columns_train = np.isnan(X_train).any(axis=0)
print("Columns with NaN values in X_train:", nan_columns_train)
X_train_no_nan = X_train[~np.isnan(X_train).any(axis=1)]
y_train_no_nan = y_train[~np.isnan(X_train).any(axis=1)]
#Model
lr = LinearRegression()

#Fit model
lr.fit(X_train_no_nan, y_train_no_nan)
#lr.fit(X_train, y_train)

#predict
prediction = lr.predict(X_test_no_nan)

#actual
actual = y_test

train_score_lr = lr.score(X_train_no_nan, y_train_no_nan)
#test_score_lr = lr.score(X_test_no_nan, y_test)
test_score_lr = lr.score(X_test_no_nan, y_test_no_nan)
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))


#Ridge Regression Model
ridgeReg = Ridge(alpha=10)

ridgeReg.fit(X_train_no_nan, y_train_no_nan)

#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(X_train_no_nan, y_train_no_nan)
test_score_ridge = ridgeReg.score(X_test_no_nan, y_test_no_nan)

print("\nRidge Model............................................\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))

plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',labe
l=r'Ridge; $\alpha = 10$',zorder=7)
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Rid
ge; $\alpha = 100$')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Li
near Regression')
plt.xticks(rotation = 90)
plt.legend()
plt.show()

#Lasso regression model
print("\nLasso Model............................................\n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train_no_nan,y_train_no_nan)
train_score_ls =lasso.score(X_train_no_nan,y_train_no_nan)
test_score_ls =lasso.score(X_test_no_nan, y_test_no_nan)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
pd.Series(lasso.coef_, features).sort_values(ascending=True).plot(kind="area")
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',labe
l=r'Ridge; $\alpha = 10$',zorder=7)

#add plot for lasso regression
#plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'l
asso; $\alpha = grid$')

#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Li
near Regression')

#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge,  and Linear regression model")
plt.show()

#Using the linear CV model
from sklearn.linear_model import RidgeCV

#Lasso Cross validation
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train_no_nan,y_train_no_nan)
```

```
#score
print("The train score for ridge model is
{}".format(ridge_cv.score(X_train_no_nan,y_train_no_nan)))
print("The test score for ridge model is {}".format(ridge_cv.score(X_test_no_nan, y_test_no_nan)))
```