WQU Marketing Handbook

Introduction

There is no doubt that we may have come across a lot of highly advanced and accurate methods to establish suitable trading strategies and apply them in real-life portfolios. Each methodology possesses its own pros and cons. As a quant, it is important to notice which methods or models to be applied in an appropriate way so that we can generate the near-accurate predicted results.

This marketing handbook provides basic and technical guides for 3 different types of machine learning models that includes classification trees, ridge regression, and K-mean clustering models.

1. Category 1: Ridge Regression

Keywords: linear regression, linear relationship, supervised machine learning, balancing trade-offs, regularization, L2 penalty, cross-validation, shrinkage, tuning parameter

1.1 Basics/Definitions and Illustration

Ridge Regression, also known as the L2 regularization method, is a type of linear regression technique that helps to reduce overfitting. It is almost identical to ordinary least squares regression, except the cost function (the penalty term) now includes. The penalty term, denoted by 'alpha,' is a regularization parameter that determines the degree of shrinkage of the regression coefficients towards zero. The greater the degree of shrinkage and the more resilient the model, the higher the value of alpha.

In general form, assuming $G(\theta)$ as the cost function of training a model with vectors of parameters θ, regularization can be achieved by adding the penalty term to the mean-squared-error (MSE) of the model, which can be expressed as follows:

$$G(\theta) = MSE(\theta) + penalty\ term(\theta)$$

Below illustration shows the relationship between model complexity and error generated. As we can see that the more or less complex the model becomes (far from its optimal complexity), the more error (bias and variance) shall be generated. While high bias occurs due to the less complexity of the model, meaning lack of enough data or important features (underfitting), high variance occurs due to the more than enough complexity of the model (overfitting). Thus, in order to avoid the underfit and overfit situations, shrinkage and regularization are applied, which is also known as bias-variance trade-offs. Further explanations shall be mentioned in the "1.3 Equation and Features" Section.

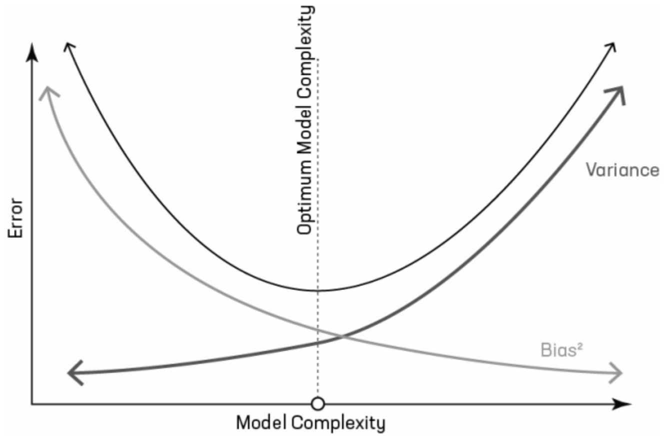

Fig 1.1: Model Complexity vs. Error | Bias-Variance Trade-offs (GeeksforGeeks)

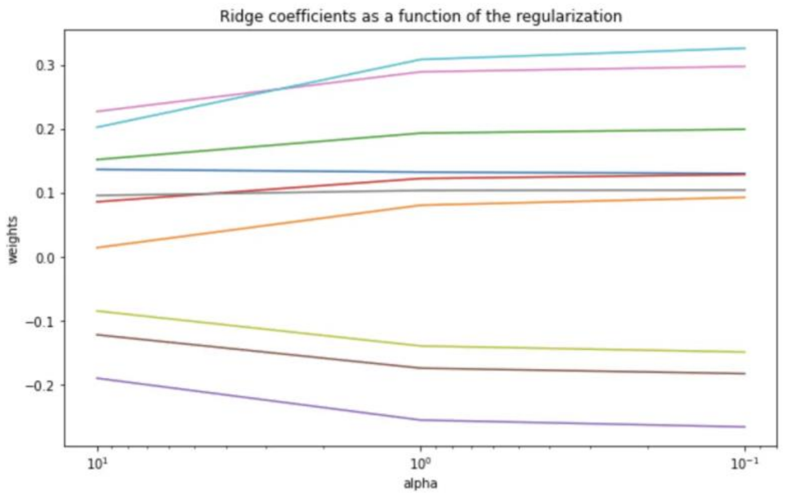

Fig 1.2: Ridge Coefficients as a function of the regularization (Kumar)

Above figure shows the example of how ridge coefficients change over different penalty terms component 'alpha'.

1.2 Advantages and Disadvantages of using Ridge Regression

As mentioned earlier, although ridge regression is very useful in situations where overfitting or multicollinearity are greatly considered, it can be a bit cumbersome for some types of data.

1.2.1 Advantages

Advantages of Ridge Regression include:

1) Handles multicollinearity: Especially for the financial data with highly correlated multiple variables, by adding a penalty term to the regression coefficients, it enables the stabilization of the estimates and improves the model accuracy.
2) Lowers Overfitting: Overfitting can occur when the model shows very low error to the training data yet performs poorly to the new data. By having the regularization term, ridge regression reduces overfitting by minimizing the coefficients toward zero. This improves the model's predictability and generalizability.
3) Handles Outliers Well: Outliers in financial data can have a substantial impact on the regression results. Ridge regression is less susceptible to outliers than traditional regression, providing more robust models.
4) Offers flexibility: Ridge regression is a tool that can handle a variety of financial problems, such as risk management, asset pricing, credit risk modeling, and portfolio optimization. It can also be used with other methodologies to generate more powerful models, such as principal component analysis.

1.2.2 Disadvantages

Disadvantages of Ridge Regression include:

1) Necessary to choose a regularization parameter: Finding an optimal value for the regularization parameter can be hard and involves appropriate tuning to balance bias and variance.
2) Assuming linear relationships: Ridge regression assumes that features (the independent variables) and target (the dependent variable) have a linear relationship, which may not be true in all financial problems.
3) Not performing feature selection: Ridge regression does not explicitly perform feature selection, which may result in a model with an excessive number of variables. This may sometimes cost the function a lot and unnecessarily increase the complexity of the model.
4) May mask important variables: Ridge regression can minimize the coefficients of some variables to zero, which can bury essential variables within the model.

1.3 Equation and Features of Ridge Regression

By taking the reference to the basic definitions of the ridge regression, it can be expressed mathematically in equations. The regularization term is

$$\alpha \sum_{i=1}^{n} \theta_i^{2},$$

where α is the hyperparameter component to consider how much regularization is considered in the model. Thus, it is quite noticeable that if α is 0, the model is not regularized at all. However, if α is very large, the data will become quite flat line and would not reflect the actual situation at all. By applying these regularization term to the general form mentioned earlier, the cost function of the ridge regression becomes

$$G(\theta) = MSE(\theta) + \frac{1}{2}\alpha\sum_{i=1}^{n} \theta_i^{\,2}$$

By taking into account that the Ordinary Least Squares (OLS) produces estimates of features coefficient $\beta_0 + \beta_1 + \beta_2 + \ldots + \beta_n$ by minimizing the $MSE(\theta)$

$$MSE(\theta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{m} \beta_j\theta_{i\,j}\right)^2,$$

The cost function will becomes

$$G(\theta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{m} \beta_j\theta_{i\,j}\right)^2 + \frac{1}{2}\alpha\sum_{i=1}^{n} \theta_i^{\,2}$$

In gradient descent form, the next step of the parameter will be given by:

$$\theta^{[next\ step]} = \theta^{[previous\ step]} - \eta\nabla_\theta\, MSE(\theta^{[previous\ step]}) - a \odot \theta^{[previous\ step]}$$

where, a is the column vectors of n numbers of elements of which have the value of α, except for the first term which shall be zero. $\odot$ means element-by-element multiplication.

A vector representation of the ridge regression is another way to look at it. This is what the ridge estimates. $\hat{\beta}_R = (X^{\ T}X + \alpha I)^{-1} X^T Y$ moves the $X^{\ T}X$ in the OLS estimates $\hat{\beta}_R = (X^{\ T}X)^{-1} X^T Y$ by (αI). It is worth noting that in the situation of "perfect" multicollinearity and $n < p$, $X^{\ T}X$ is singular (non-invertible), and shifting $X^{\ T}X$ by (αI) results in an invertible matrix $(X^{\ T}X + \alpha I)$. (Du)

Summarizing the important features of ridge regression include:

1) Shrinkage of coefficients
2) Bias-variance tradeoff
3) Multicollinearity
4) Ridge Path
5) Computational Efficiency

1.4 Guide

Here are the lists of inputs and outputs of ridge regression:

1.4.1 Inputs

1) A set of input features θ: A matrix of size $(n \times p)$, where n is the number of observations and p is the number of predictor variables.
2) A set of output values y: A vector with size (n) representing the target variable.

1.4.2 Outputs

1) Estimated Coefficient β: A vector with size (p), representing the coefficient estimates of the regression model.
2) Regularization parameter α: A scalar value that determines the strength of the penalty term added to the function.
3) Predicted values of $\hat{y}$: A vector with size (n), representing the predicted values of target variables for the input features θ
4) Goodness of fit measures, e.g., R^2, MSE, CVE: Metrics to evaluate how well the ridge regression model fits the data by quantifying it. (CVE = Cross-Validation Error)

1.5 Hyperparameters

Here is the list of hyperparameters of ridge regression that need tuning by the user or chosen through a search process.

1) Regularization parameter α: The value of α is usually selected by cross-validation to minimize the test error. It can range from 0 (no regularization) to ∞ (infinite regularization - flatline)
2) Scaling: Ridge regression can be sensitive to the scaling of the input features. Thus, it is often beneficial to normalize or scale the input features to have a mean of 0 and unit standard deviation.
3) Degree of polynomial features: In order to improve the performance and robustness of the model, ridge regression can be extended to include polynomial features of the original input features. This complexity can help involve nonlinear relationships in the function.

1.6 Computation of Ridge Regression in Python Notebook (Google Colab|Jupyter)

In Python language, in order to compute the ridge regression, the scikit-learn module is required to be imported into our notebook. Below are the codes for necessary modules:

```
`from sklearn.linear_model import Ridge`
`from sklearn.model_selection import train_test_split`
```

`from sklearn.metrics import mean_squared_error`

`Ridge` is the class that implements Ridge regression in scikit-learn. `train_test_split` is a commonly used function for splitting the dataset into training and testing sets. `mean_squared_error` is a function to compute the mean squared error of a regression model to determine the goodness of the model.

Another parameter that is important to consider when using `Ridge` module is the value of `alpha` which is the regularization parameter α. For example,

`ridge = Ridge(alpha=0.5)`

In this example code, α value is taken as 0.5.

The details about computation with ridge regression in the Python Notebook is attached separately from this handbook. The dataset used in the Notebook ‘housing.csv’ is taken from the 1990 California census . The information refers to the housing dataset and related summary stats.

The dataset is tiny, but its complexity derives from its enormous multicollinearity to predict the mean house values

2. Category 4: Classification trees

Keywords: decision tree, predictive modeling, non-parametric, tree pruning, feature selection, interpretability, nodes, branches

2.1 Basics/Definition and Illustrations

Classification tree is a supervised learning algorithm that makes predictions on input by going through a tree-like structure of binary decisions. The algorithm splits the training data into subsets by recursively considering the values of the features until the subsets are as pure as possible in terms of class labels. The resulting tree can be used to classify new data points according to their feature values. The tree structure is dependent on the feature and threshold chosen. Ideally, the feature that best separates the data should be selected. These conditions ensure that the tree ultimately converges towards final leaves with minimized uncertainty. Below is an example of a classification tree.

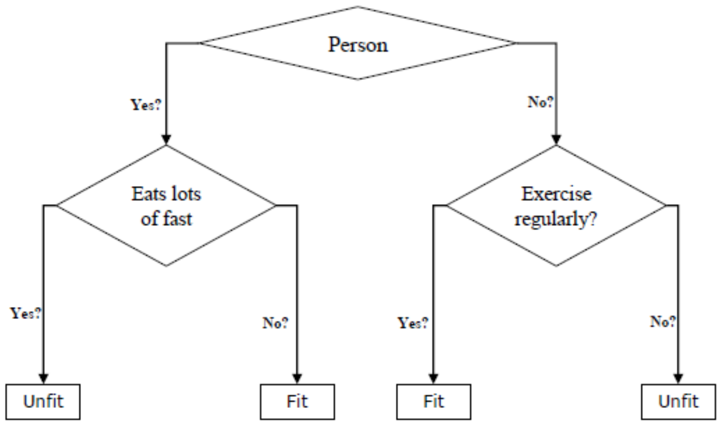

Fig 2.1: Simple Classification Decision Tree (TutorialsPoint)

2.2 Advantages and Disadvantages of Classification trees

Classification trees method also possesses beneficial factors or cumbersome facts when dealing with certain types of data and situations.

2.2.1 Advantages

Advantages of classification trees include -

1) Easy to interpret and comprehend due to their visual nature. They provide a transparent decision-making process that can be communicated effectively to stakeholders who lack technical expertise.
2) Can work with both categorical and continuous data, making them a versatile tool for a wide range of data types.
3) Non-parametric, meaning that they do not make assumptions about the data distribution, which makes them robust to outliers and skewed data.
4) Can handle missing data by simply ignoring them and making decisions based on the available data.
5) Fast and scalable, allowing them to work well with large datasets.

2.2.2 Disadvantages

Disadvantages of classification trees include -

1) Not equipped to handle continuous target variables, as they are designed solely for classification tasks.

2) Tendency to overfit, particularly when the tree is deep and has many nodes. One way to address this issue is through tree pruning, which involves removing branches that do not enhance the accuracy of the tree.
3) Sensitive to the choice and representation of features. For instance, categorical features with numerous values can lead to an extensive number of branches, making interpretation difficult.

2.3 Equations and Features

Classification trees use a decision tree structure to recursively split the data based on the values of the features. The splits are chosen to maximize the separation between the classes of the target variable. There are no specific equations associated with classification trees, as the algorithm works by recursively splitting the data based on feature values, and determining the best feature and threshold to use at each node in the tree.

Some important features of classification trees include:

- Feature selection: Choosing the best feature and threshold to split the data on at each node in the tree
- Tree pruning: Removing branches that do not improve the accuracy of the tree in order to prevent overfitting
- Handling missing values: Ignoring missing values and making decisions based on available data
- Non-parametric: Not making assumptions about the distribution of the data
- Interpretability: Providing a clear and easy-to-understand decision-making process, especially when visualized
- Handling both categorical and continuous data: Being able to work with a wide range of data types.

2.4 Guide

The inputs to a classification tree algorithm are typically a dataset of labeled examples, where each example is represented as a vector of features and a corresponding class label. The algorithm uses this dataset to learn a decision tree that can predict the class label of new, unseen examples based on their feature values.

The output of a classification tree algorithm is the decision tree itself, which is a structure consisting of nodes and branches. Each node in the tree corresponds to a binary decision based on the value of a particular feature, and each branch represents the possible outcomes of that decision. At the leaf nodes of the tree, the algorithm assigns a predicted class label to any input that reaches that point in the tree.

Additionally, the algorithm may also output some metrics related to the performance of the decision tree, such as its accuracy, precision, recall, or F1 score, which can be useful for evaluating and comparing different models.

2.5 Hyperparameters

Hyperparameters are settings that are not learned from the data, but rather are specified by the user or chosen through a search process. In classification trees, there are several hyperparameters that can be adjusted to affect the behavior and performance of the model. Some common hyperparameters of classification trees include:

1) Maximum depth: This limits the depth of the decision tree by restricting the number of levels that can be added to the tree. Setting a smaller maximum depth can help prevent overfitting, but may result in lower accuracy.
2) Minimum number of samples per leaf: This sets a minimum threshold for the number of samples required to create a leaf node. Increasing this value can prevent the model from splitting too much on small subsets of the data, but may result in less specific decisions.
3) Minimum number of samples to split: This sets a minimum threshold for the number of samples required to split a node. This hyperparameter can help prevent the model from overfitting on noise, but may result in a less expressive tree.
4) Split criterion: This determines the measure of impurity used to determine the best split. The two most common criteria are Gini impurity and entropy. Gini impurity is faster to compute and tends to create smaller trees, while entropy tends to create more balanced trees.
5) Tree pruning: This involves removing branches of the tree that do not improve its accuracy. Pruning can help prevent overfitting and improve the model's performance on new data.

2.6 Computation of classification trees in Python

In Python language, in order to compute the classification tree, the scikit-learn module is also required to be imported into our notebook. Below are the codes for necessary modules:

```
`from sklearn.tree import DecisionTreeClassifier`
`from sklearn.model_selection import train_test_split, GridSearchCV`
`from sklearn.metrics import classification_report, confusion_matrix`
```

```
`from sklearn import tree`
`from sklearn.metrics import roc_auc_score, roc_curve`
`from sklearn.metrics import mean_squared_error`
```

This time, we use `` `DecisionTreeClassifier` `` to train classification tree models for the provided dataset in order to predict whether income exceeds $50K/yr based on census data. `` `GridSearchCV` `` is a cross-validation search model. We also additionally use `` `classification_report, confusion_matrix, roc_auc_score, roc_curve` `` metrics to know about the goodness of the fit of the model to the dataset.

The details about computation with classification trees in the Python Notebook is attached separately from this handbook.

3. K-means Clustering:

Keywords: K-means, clustering, centroids, distance, iterations, euclidean, inertia, elbow method, inertia, Silhouette score, cluster analysis

3.1 Basics/Definitions and Illustration

K-means clustering is a popular unsupervised machine learning approach for grouping or clustering unlabeled data points based on their similarity. Here's a K-means clustering manual that includes all of the features you mentioned.

K denotes the number of predefined clusters that must be produced during the procedure; for example, $K = 2$ means there will be two clusters; $K = 3$ means there will be three clusters, and so on.

It is a centroid-based method, with each cluster having its own centroid. This algorithm's main goal is to minimize the sum of distances between data points and their respective clusters. (JavaTPoint)

Below illustration shows how the K-means clustering algorithm works to group the unorganized scattering data points into data clusters with similar characteristics or features.

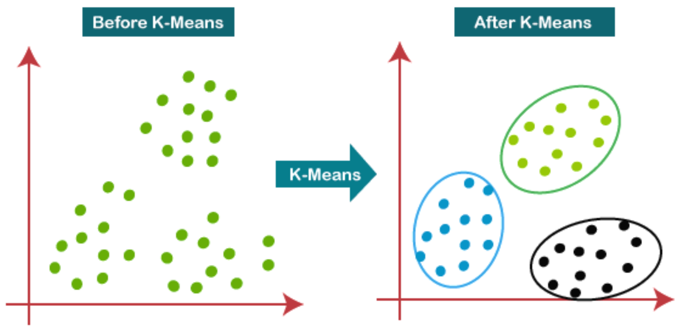

Fig 3.1: K-means algorithm working (JavaTPoint)

3.2 Advantages and Disadvantages of K-means clustering

Unlike the above two methods, ridge regression and classification tree, K-means clustering is the most commonly used algorithm for unsupervised machine learning, especially for the dataset with unlabeled data and predetermined target variables.

3.2.1 Advantages

Advantages of K-means clustering include -

1) Computationally efficient: K-means clustering often converges quickly to a stable solution, reducing computing workload and making it suitable for real-time applications.
2) Easy to understand and implement: This is due to the nature of being a simple and intuitive algorithm that requires minimal parameters.
3) Scalability: K-means clustering is a fast and scalable algorithm that works well with huge datasets containing a lot of variables and observations.
4) Robust to outliers: Since K-means clustering uses centroid and mean distance as the center of computation, outliers shall have less impact on the overall model, compared to other models.
5) Interpretability: K-means clustering creates clusters that are simple to understand and have distinct boundaries that can be seen and examined.

3.2.2 Disadvantages

Disadvantages of K-means clustering include -

1) Sensitivity to initial conditions: The initial selection of centroids can have a significant impact on the quality of the clustering result, which can result in undesirable outcomes.
2) Dependency on K value: K-means clustering requires a predetermined number of clusters, and sometimes, choosing the right number of clusters can be challenging.
3) Sensitivity to scaling: Since K-means clustering uses distance metric to determine the similarity between observations, scaling can alter the distance, which makes the model less accurate.
4) Only spherical clusters: K-means clustering presumes spherical clusters with equal variances, which may not necessarily be the case in real-world datasets.
5) Inability to handle complex relationships and data overlapping: K-means clustering is a linear technique that cannot capture complex nonlinear interactions between variables, usually found in real-world datasets.

3.3 Equations and Features

K-means clustering works by classifying a dataset into a predetermined number of clusters. Below steps are the explanation on how the algorithm works in mathematical form.

- Step-1: Firstly select the K, number of clusters
- Step-2: Randomly place the K centroids at different random places
- Step-3: Then, from each point in training data, calculate the distance to each centroid
- Step-4: These points are grouped to the nearest centroid.
- Step-5: Calculate means of each group and reassign the centroids to the location of that means.
- Step-6: Repeat the steps 3 to 5 until optimal number of K has reached.

The flowchart of these steps will be as follows:

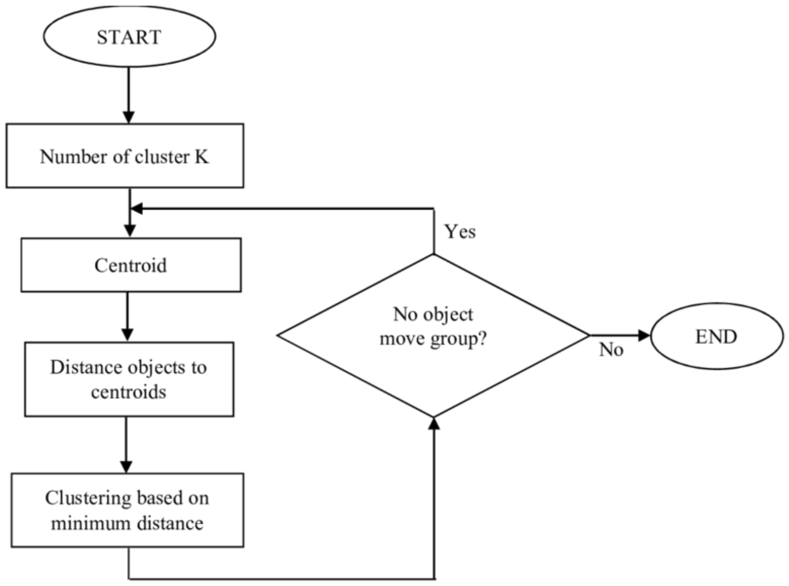
START
Number of cluster K
Centroid
Distance objects to centroids
Clustering based on minimum distance
No object move group?
Yes
No
END

Fig 3.2: Flowchart of K-means clustering algorithm (Bustamam)

Mathematically, the purpose of K-means clustering is to minimize the objective function which is the squared error function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} || {x_i}^j - C_j ||^2$$

where x_i is the i-th data point, C_j is the j-th centroid and k is the number of clusters.

Besides, for the above mentioned steps, mathematical expression is given by assuming $X = \{x_1, x_2, \ldots, x_n\}$ be our data points and $V = \{v_1, v_2, \ldots, v_c\}$ be the centroids, the mean of the clustering can be calculated by using the following equation

$$v_i = (\frac{1}{c_i}) \sum_{j=1}^{c_i} x_j$$

where c_i is the number of data points in the i-th cluster.

Some important features of K-means clustering include:

- Centroids: Based on the distance between data points and centroids, K-means clustering need to identify the fixed number of centroids.
- Euclidean distance: Direct distance between data points and centroid in the same cluster, used to determine the similarity.
- Iterations: K-means clustering is the iterative methods that need to keep updating the position of centroid until the optimal value of K reach.
- Convergence criterion: Algorithm usually stops when there is no more movement of centroids or when the maximum number of iterations has reached.
- Elbow method: Used to determine the optimal number of clusters
- Silhouette Score: a quality measure of the clustering result that considers both the distance between observations and their assigned centroids as well as the distance between observations and the centroids of the nearest neighboring cluster.
- Initialization: Required initial choice of centroids which can be randomly generated or based on prior knowledge.

3.4 Guide

The K-means clustering technique takes as input a dataset with n observations and m variables/features, which is represented as a matrix X with dimensions $(n \times m)$. The algorithm additionally requires a parameter k, which indicates the number of clusters to be created.

The K-means clustering algorithm produces a set of k clusters, with each observation assigned to the nearest centroid based on the distance measure. The following are the results:

- Cluster Assignments
- Centroids
- Inertia
- Convergence Status

The cluster assignments can be used to group together comparable observations and find patterns or subgroups in the data. The centroids can be used to depict and comprehend the clustering result by representing the typical characteristics of each cluster. The inertia can be used to compare the quality of different clustering outcomes, and the convergence status can be used to evaluate whether the method should be run again.

3.5 Hyperparameters

Some hyperparameters of the K-means clustering include -

1) Number of clusters (k): This is the most significant hyperparameter, specifying the number of clusters to be generated.
2) Initialization technique: It can impact the quality of the clustering result.
3) Max: iteration numbers: It determines how many times that iterations shall be continued before terminated.
4) Choice of distance metrics: It measures the similarity between data points and centroids. For e.g. the Euclidean distance, Manhattan distance or cosine similarity.
5) Convergence threshold: It determines the minimum change in the position of centroids required to continue the iterative process.
6) Cluster Initialization Method: Different methods like random or K-means++ can be used.

3.6 Computation of K-means clustering in Python

In Python language, the scikit-learn module is used to compute K-means clustering. Below are the codes for some necessary modules:

```
`from sklearn.cluster import KMeans`
`from sklearn.datasets import make_blobs`
```

This time, `` `KMeans` `` is used to compute K-means clustering and `` `make_blobs` `` is used to generate the random clusters with specific numbers of points and clusters, together with centroids. Besides, we use WCSS (Within-Cluster Sum of Square) | Elbow method to find the optimal value of k -clusters.

4. Technical Section

Here, we discuss the process and functions used to code in the Jupyter Notebook.

4.1 Ridge Regression

After performing basic exploratory data analysis and cleaning the data, the model was trained by using the target as ‘mean_house_value’ and features variables as the rest. Null values from 'total_bedrooms' column are amputated with column median value for more accurate results. Then, the initial generated results show "r2 scores ~ 0.64369" and "bias(intercept_) ~ -3573489.604". We can find that the intercept_ term is quite large, meaning the model is still underfit though the r2 score is 64.35%.

In order to know more about alpha value, cross-validation is performed by using **`RepeatedKFold`**, **`RidgeCV`** and **`GridSearchCV`**. The initial CV process also generated the mean absolute error (MAE) as -51221.171, which is the largest and best alpha return as almost one. This suggests some important hyperparameters like features need tuning. The huge MAE value also suggests that our data will need to be scaled. We used **`MinMaxScaler`** to perform scaling the range.

After standardizing the data, the CV process is performed for the next time. This time, we can see the much better value of MAE and alpha value close to zero, which means the model performs almost identical to the linear regression OLS without regularization parameter.

4.2 Classification Tree

After performing basic EDA, since our objective is to classify the person with an income level who earns above 50,000 USD a year, the target variable is created accordingly. Below is the code:

`data['target'] = np.where(data.target == ' <=50K', 0, 1)`

After that, we will select four features to demonstrate the classification tree. These features are 'age', 'education_num', 'sex' and 'hours_per_week'. With them, we will try to build the classification tree of the target. Before starting the tree algorithm, there is one selected feature with object type i.e., 'sex'. In order to become computation competitive, we will encode it with Male as 1 and Female as 0.

The dataset is split into two, train and test data by using **`train_test_split`**. After that, a decision tree classifier object is generated by using the hyperparameters of maximum depth as 3, minimum samples split as 5, andminimum samples leaf as 5. Following is the command used,

`clf = DecisionTreeClassifier (criterion="log_loss", max_depth=3, min_samples_split=5, min_samples_leaf=5)`

4.3 K-means Clustering

For this method, we take the sample dataset reference from GitHub. The information is about the Mall Customer and it contains five different variables,
"CustomerID", "Gender", "Age", "Annual Income (k$)" and "Spending Score (1-100)".

After performing basic EDA, we find that there is no null value in the dataset and it has the total observations of 200 Customers. For easy visualization, we will use the "Annual Income (k$)" and "Spending Score (1-100)" variables and drop any duplicates. This could achieve by using the command,

```
`newdf_km = df_km.copy()`
`newdf_km.drop_duplicates(inplace=True)`
`Xv = newdf_km.iloc[:, [2, 3]].values`
```

In that way, Xv becomes the new dataset with the desired features to group the clusters

Before starting the K-means clustering, we need to determine the most important hyperparameter, i.e., the value of k. In order to do so, we will use WCSS, Elbow method to determine the optimal value. This could achieve by using the following command,

```
`wcss = []`
`for i in range(1, 11):`
    `k_means = KM(n_clusters = i, init = 'k-means++', random_state = 1)`
    `k_means.fit(Xv)`
    `wcss.append(k_means.inertia_)`
```

After graphing the WCSS, we notice that after $k = 5$, the graph becomes smooth. Thus, we will choose 5 as our value of k. Moreover, regarding the number of iterations, it is unlikely to have enormous processes since the data size is small. Thus, we will not limit the maximum number of iterations in our command.

Detailed information and code are included in the Python notebook, attached separately.

5. Marketing Alpha

In this section, the computed results are mentioned in accordance with each individual report.

5.1 Ridge Regression

Although the data provided are not actually clean, after a bit of preprocessing on data, we can find that the model performs good enough on test data as well. The dataset contains 9 features and 1 target variable. Many preprocessing steps including correlation matrix were performed to get the rough idea about multicollinearity of the data. Apart from the "ocean_proximity" feature, others are in the form of numeric. Hence, Label Encoding is performed on "ocean_proximity" to convert into numeric value based on count density.

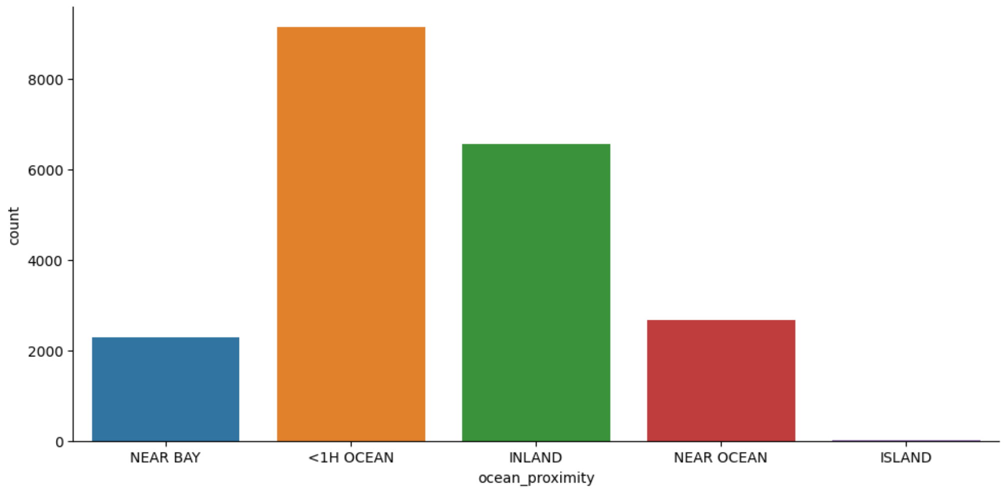

Fig 5.1: Plot of Label Encoding on “ocean_proximity” (Notebook)

When training the model for the first time, we found that though the R2 score shows more than 64%, there was a very huge bias value, which could lead to underfitting of the model. Thus, data manipulation like Min-Max-Scaling and cross-validation are performed respectively.

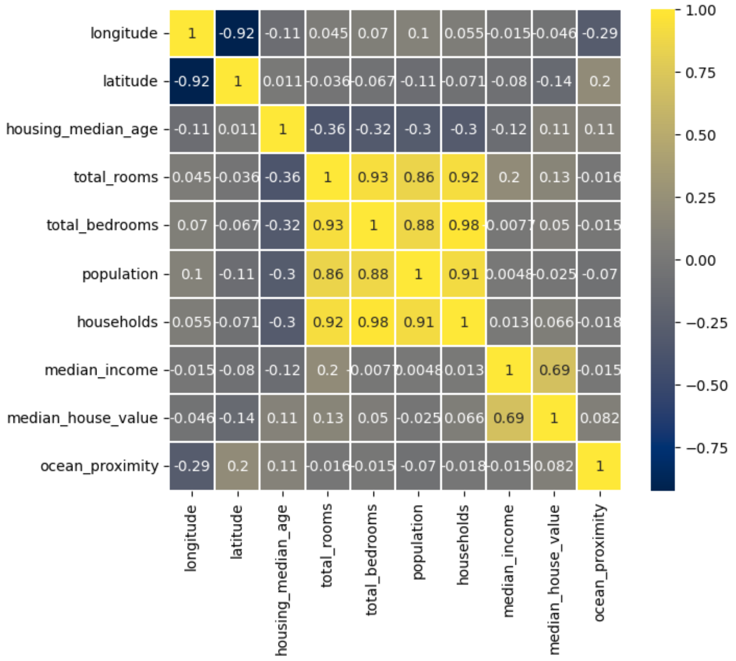

Fig 5.2: Correlation Matrix between different variables (Notebook)

Noticing the yellow area at the center of the matrix, we decided to add another variable called (rooms per households) to counter the high correlation.

After rescaling the data, we again trained the model and applied it to the test data. This time, we received the very good metrics of

- Mean-squared Error : 0.0201
- Root Mean-Squared Error : 0.14178
- R2 Scores : 0.6425

respectively. In order to understand more about the prediction and actual, some figures were plotted.

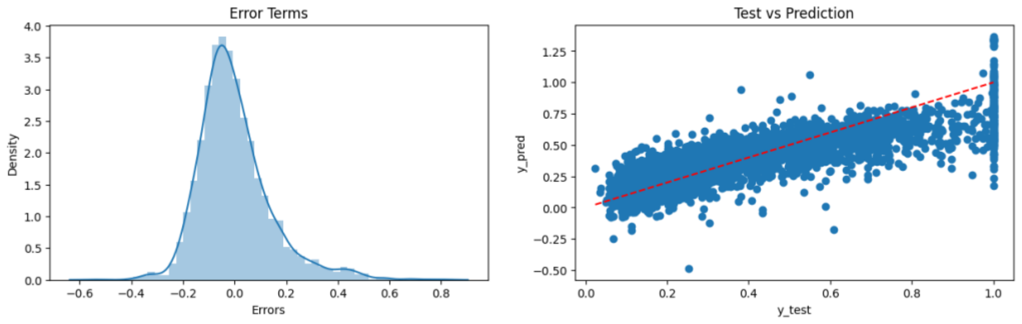

Fig 5.3: Error term and y_pred vs y_test scatter plot (Notebook)

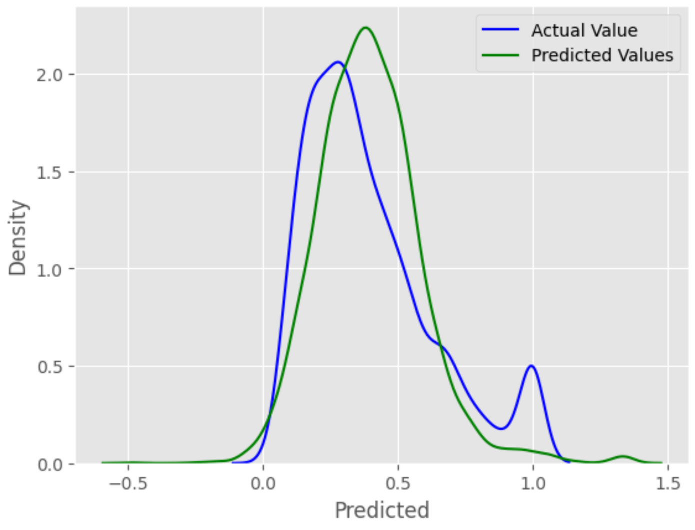

Fig 5.4: Actual value of y_test vs. predicted value of y_pred (Notebook)

Apart from the notes mentioned here and jupyter notebook, the model has good and accountable features of prediction accuracy and computational efficiency. Depending on requirements, the model enables more complexity by allowing polynomial terms in the dataset input, which inturn makes the model to have less linear relationship with other features variables.

5.2 Classification Tree

Information about the input dataset - Predict whether income exceeds $50K/yr based on census data. The dataset used is US Census data which is an extraction of the 1994 census data which was donated to the UC Irvine's Machine Learning Repository. The data contains approximately 32,000 observations with over 15 variables. The dependent variable in our analysis will be income level and who earns above $50,000 a year.

When we run the model using maximum depth of 3, the AUC-ROC metrics is 79.3%.

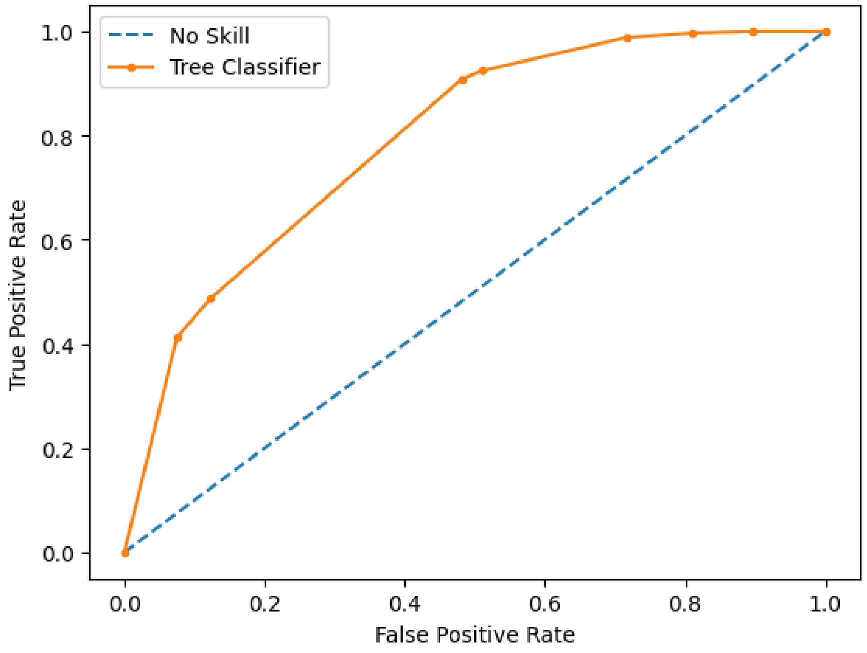

Fig 5.5: TPR vs. FPR graph

The classification tree is shown below:

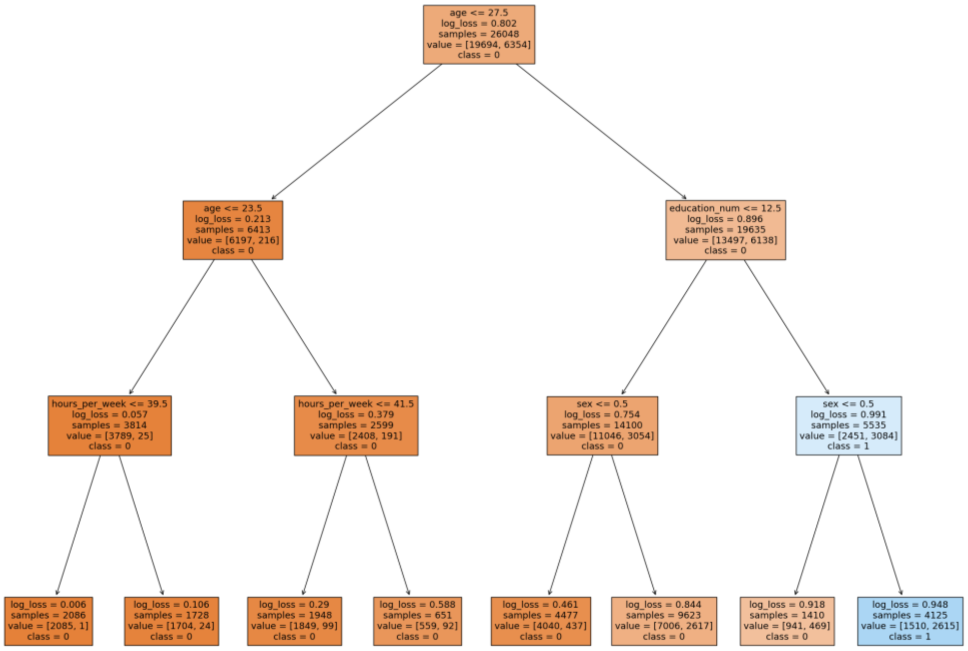

Fig 5.6: Classification Tree Output

5.3 K-means Clustering

The reference dataset used in this computation is tiny. However, it covers the foundation level process of K-means clustering. (Castagno)

The generated Elbow curve is as below,

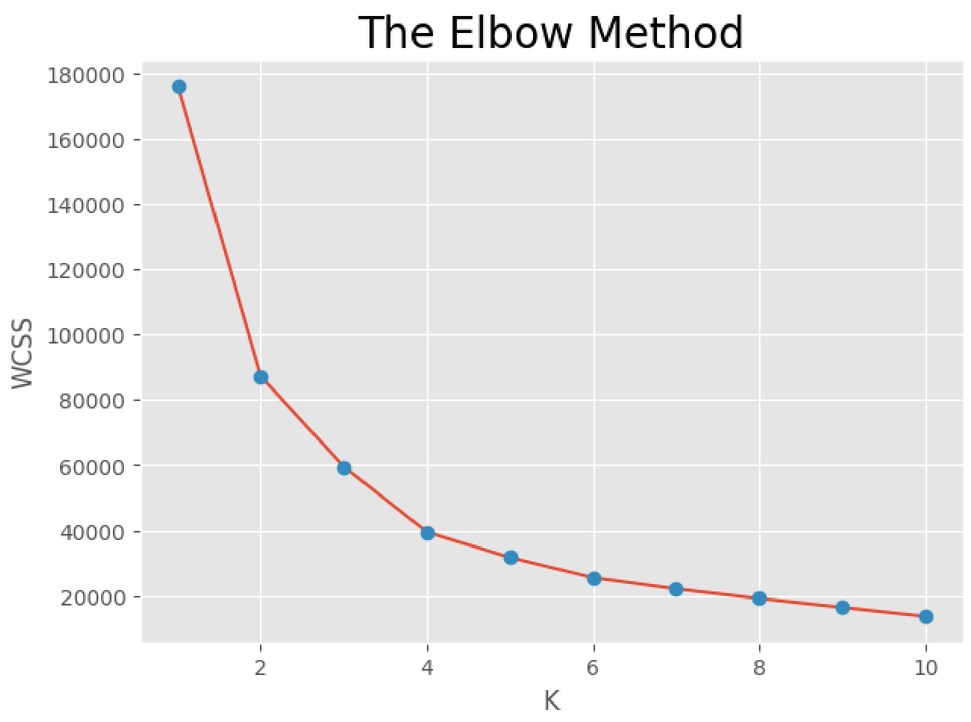

Fig 5.7: The Elbow Curve

By looking at the curve, we notice that after $k = 3$, the slope of the curve significantly reduced. Besides, after 5, the slope becomes very smooth. Thus, we will use 5 as the number of clusters to further group the similar data into clusters. While doing the K-means clustering, we regard some hyperparameters of the initialization method as “k-means++”, and “random_state” as 1.

Below data shows the finalized 5 clusters with their respective centroids, based on the “Annual Income (k$)” and “Spending Score (1-100)” variables.

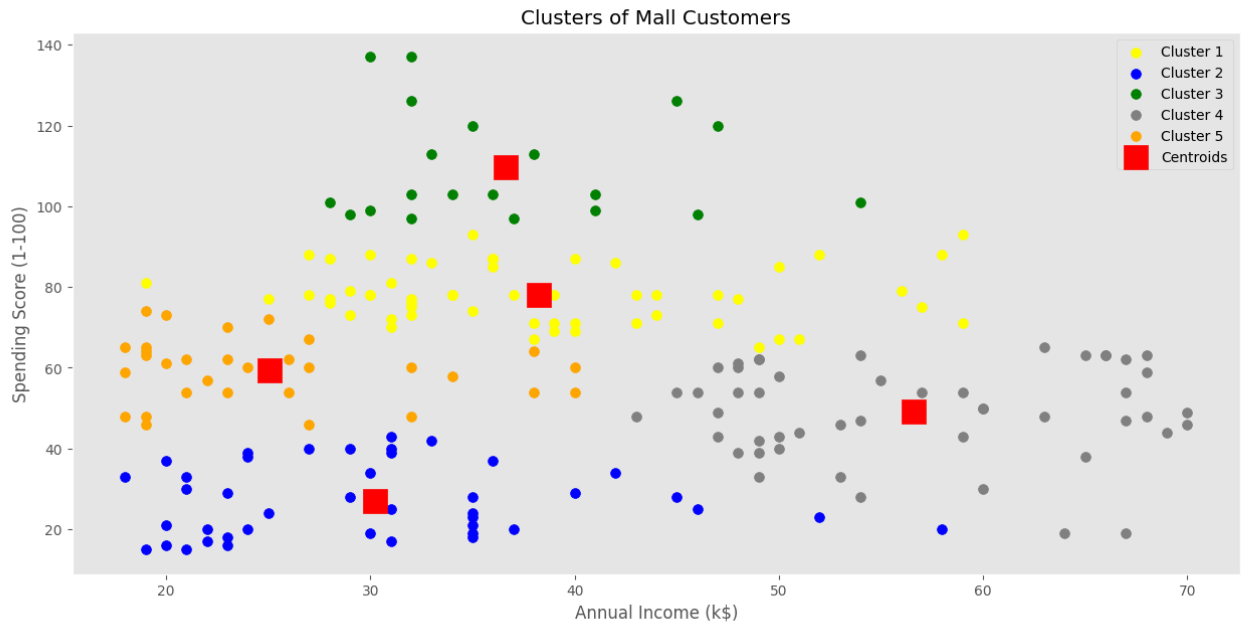

Fig 5.8: Clusters of Mall Customers using K-means clustering algorithm

6. Conclusion

To summarize, machine learning is a rapidly expanding area that has altered how we approach data analysis and decision-making. It has allowed us to extract insights from big and complicated information, automate procedures, and build predictive models to increase the accuracy and efficiency of numerous operations. During the first three weeks with the Machine learning course, although there are still many complex and useful algorithms left to explore and learn, we believe that ML shall become more common and powerful, creating new opportunities and challenges for the businesses and society as a whole.

Cited Journals and Learn More

Below are the cited references and journal articles that are referenced in this handbook in MLA format.

1) Banfield, Robert & Hall, Lawrence & Bowyer, Kevin & Kegelmeyer, W.. (2007). A Comparison of Decision Tree Ensemble Creation Techniques. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 29. 173-180. 10.1109/TPAMI.2007.250609.
2) Brownlee, Jason. "How to Develop Ridge Regression Models in Python - MachineLearningMastery.com." Machine Learning Mastery, 9 October 2020, https://machinelearningmastery.com/ridge-regression-with-python/. Accessed 4 April 2023.
3) Bustamam, A. et al. "Application of K-means Clustering Algorithm in Grouping the DNA Sequences of Hepatitis B Virus (HBV)." AIP Conference Proceedings, vol. 1862. no. 1. AIP Publishing LLC, 2017.
4) Castagno, Patrizia. "k-Means Clustering (Python)." medium, 27 December 2022, https://medium.com/@patriziacastagnod/k-means-clustering-python-4eec18b4f0ec. Accessed 5 April 2023.
5) Du, Dr. Hailiang. "Machine Learning and Neural Networks." Bookdown, 06 12 2022, https://bookdown.org/hailiangdu80/Machine_Learning_and_Neural_Networks/shrinkage-methods.html. Accessed 4 April 2023.
6) Du, Xingqiang, Wei Wang and Rongfang Bie. "A decision tree-based approach for predicting customer churn in cellular network services", journal Expert Systems with Applications, 2016
7) GeeksforGeeks. "Lasso vs Ridge vs Elastic Net | ML." GeeksforGeeks, 10 January 2023, https://www.geeksforgeeks.org/lasso-vs-ridge-vs-elastic-net-ml/. Accessed 4 April 2023.
8) Hoerl, Arthur E., and Robert W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." Technometrics, vol. 42, no. 1, 2000, pp. 80–86. JSTOR, https://doi.org/10.2307/1271436. Accessed 5 Apr. 2023.
9) JavaTPoint. "K-Means Clustering Algorithm." Javatpoint, https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning. Accessed 5 April 2023.
10) Konstantinidis, Stavros N. and Karkanis, Ioannis D. "Decision tree induction in medical diagnosis: a case study" journal Artificial Intelligence in Medicine, 2007, Accessed 2 April 2023,
11) Kumar, Narender. "Ridge Regression With Examples." Spark By {Examples}, 2 April 2023, https://sparkbyexamples.com/machine-learning/ridge-regression-with-examples/. Accessed 4 April 2023.

12) TutorialsPoint. “Classification Algorithms - Decision Tree.” Tutorialspoint, https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_decision_tree.htm. Accessed 3 April 2023.

13) WorldQuant University. “MACHINE LEARNING IN FINANCE MODULE 1 | LESSON 2.” SUPERVISED MODELS: REGRESSION AND HYPERPARAMETERS, WQU, Accessed 4 April 2023.