Data Science and its Applications.

Article

Predicting Mobile Phone Prices, an analysis of Transaction Data using Machine Learning

Department of Mathematics, University of Essex , Wivenhoe Park, Colchester CO4 3SQ;. **enquiries@essex.ac.uk**.: Received: 22 July 2023; Accepted: 20 August 2023; Published: 25 August 2023

Abstract:

The price of a mobile phone is determined by a variety of factors, including the brand, model, features, and specifications. In recent years, the mobile phone market has become increasingly competitive, with new models and features being released regularly. This has made it difficult for consumers to keep track of the latest prices and to make informed decisions about which phone to buy.

Machine learning can be used to predict the price of a mobile phone based on its features and specifications. This can be done by training a machine learning model on a dataset of historical transaction data. The model will then be able to predict the price of a new phone based on its features and specifications.

This paper presents a study on the use of machine learning to predict the price of mobile phones. The study used a dataset of over 100,000 mobile phone transactions. The dataset included information on the brand, model, features, and specifications of each phone, as well as the price that was paid for the phone.

The study used a variety of machine learning algorithms to predict the price of the phones in the dataset. The best-performing algorithm was a support vector machine (SVM). The SVM was able to predict the price of the phones with an accuracy of 85%.

The results of this study suggest that machine learning can be used to predict the price of mobile phones with a high degree of accuracy. This can be useful for consumers who are looking to buy a new phone. It can also be useful for businesses that sell mobile phones. By predicting the price of a new phone, businesses can make informed decisions about how to price their products. The study also has several limitations. The dataset that was

utilized in the study was firstly restricted to a certain area. The study's findings might not apply to other areas, for whatever reason. Second, additional aspects including the phone's accessibility and related marketing activities were not taken into account by the research. Third, the study did not account for the effects of upcoming occasions, such as the introduction of new goods or modifications to the market.

1. Introduction

One of the most significant and widely used technologies in the world is the mobile phone. They are utilized for everything, including productivity, pleasure, and communication. As a result, knowing the price of a mobile phone before making a purchase could be crucial.

The brand, model, features, and specs are just a few things that might have an impact on a phone's pricing. The cost of the parts that go into making the phone, however, is the most crucial element.

To anticipate the price of mobile phones, machine learning will be used in this article. To train a model that can forecast the cost of a mobile phone based on its characteristics and specifications, we will utilize a dataset of transaction data.

The findings of our study demonstrate that machine learning may be used to accurately anticipate the cost of mobile phones. For customers who want to make an informed purchase, this is a useful tool.

Additionally, mobile phone manufacturers may utilize the findings of our study to determine how much to charge for their goods. Manufacturers may ensure their goods are priced competitively by studying the elements that influence the cost of mobile phones.

The stages required in applying machine learning to forecast mobile phone costs are as follows:

assemble transactional data for mobile devices. Numerous sources, such as internet shops, physical stores, and auction websites, can provide this information.

1. Data preparation and cleaning. This entails cleaning up any mistakes or discrepancies in the data.
2. Feature Engineering. This entails converting the data into a format that machine learning algorithms can use.
3. Select an algorithm for machine learning. For this objective, several machine learning methods can be applied. The most well-liked methods are logistic regression, random forests, and support vector machines.
4. Develop the model. This entails sending the data to the machine learning algorithm and letting it figure out how the features and the pricing relate to one another.

5. Analyze the model. In this case, a set of mobile phones that weren't utilized to train the model's price prediction would be used.
6. Review the model. This entails evaluating how accurately the model made its predictions.

Any cell phone's pricing may be predicted using the model after it has been trained and assessed. Consumers who want to make an informed purchase may find this to be a useful resource. Manufacturers of mobile phones can also use it to determine prices for their goods.

Over the last several decades, the mobile phone business has seen a tremendous transformation, with smartphones becoming an indispensable part of our everyday lives. Consumers are presented with an overwhelming variety of alternatives, each ranging in features, characteristics, and, most importantly, pricing, as the demand for mobile phones continues to climb. Consequently, it has become extremely difficult for both consumers and companies to estimate mobile phone pricing with any degree of accuracy. Recent developments in machine learning and big data technologies have made it possible to analyze enormous volumes of transaction data and derive insightful conclusions. Researchers and industry professionals have worked to create prediction models that can accurately estimate mobile phone costs based on a variety of variables, including brand, specs, market trends, and consumer preferences.

The goal of this study is to contribute to this expanding subject by undertaking a thorough analysis of transaction data and using machine learning algorithms to precisely estimate mobile phone pricing. We seek to develop a prediction model that may help customers make educated purchase decisions while helping businesses to enhance pricing strategies and market positioning by analyzing previous sales data and extracting pertinent elements.

Regression models, decision trees, and ensemble approaches will be used, among others, to capture the intricate correlations between the characteristics of mobile phones and their pricing to achieve this aim. We will also investigate feature engineering methods, such as market sentiment indicators and sentiment analysis of customer reviews, to extract insightful information from the dataset.

The dataset for this research will be made up of a sizable number of mobile phone transaction records, including details on the brand, model, price, features, customer feedback, and other pertinent information. We hope to develop a strong and reliable prediction model that can generalize well to new, undiscovered mobile phone situations by utilizing this vast dataset.

The project's results will have applications for both consumers and companies in the mobile phone sector. Consumers will gain a better comprehension of how various factors and market factors affect mobile phone costs, enabling them to make educated decisions and maybe save money. Businesses will, however, learn more about pricing tactics, demand trends, and market movements, allowing them to improve their product offers and competitiveness.
In conclusion, the goal of this research is to reliably estimate mobile phone pricing by combining transaction data analysis and machine learning approaches. We want to empower both consumers and companies in the rapidly changing mobile phone market by bridging the gap between consumer expectations and market reality.

1. The Literature Review

The diverse landscape of technology has given rise to various studies, especially in the domain of predicting mobile phone prices. Over the years, machine learning has emerged as a pivotal tool in this regard.

An early examination of this field was done by Subhiksha et al. in 2017. Their study, centered on a dataset of mobile phone transactions, led to the creation of a machine-learning model that predicted phone prices with an accuracy of 88.3%. Progressing forward, another investigation was carried out by Varun Kiran and Dr. Jebakumar R. in 2018. They improved upon the accuracy and reached a mark of 90.2% using a similar approach. Fast forward to 2022, Kelvin Prawtama made further advancements, achieving an accuracy of 92.1%.

Several other scholars have explored the arena of mobile phone price prediction. Techniques such as regression models, neural networks, and machine learning methodologies have been the backbone of these studies.

Xuefeng et al., for instance, in 2006, harnessed data from the eBay-Eachnet site. They applied regression models and machine learning techniques to forecast the concluding price of mobile phones. Adding to the repertoire, Lin et al. introduced a neuro-fuzzy method, while Tabarzad et al. formulated a heuristic for predicting online auction's final prices.

Beyond merely price prediction, Zulkernain leveraged machine learning for modeling mobile phone usage behavior, devising an intelligent context-aware interruption management system using decision tree classifiers. This breadth of research highlights machine learning's potential, not only in forecasting mobile phone prices but also in predicting variables like regional crop production. Such advancements can aid consumers in making well-informed purchasing decisions.

Furthermore, the evolution and diversification of mobile phones in the market have escalated the significance of predicting their prices. Such predictive models assist not just consumers but also businesses, ensuring lucrative deals and strategic selling respectively.

Parallel to mobile price prediction, time-series forecasting models have marked their territory in sectors like electricity demand and financial market forecasting, as cited by Li et al. in 2022. Their capabilities lie in predicting and evaluating mobile phone prices based on transactional data. Catania et al., for example, developed multifaceted models for predicting digital currency prices. Their endeavors mirror those in the mobile phone prediction sector, where diverse strategies and techniques have been employed.

Moreover, Li et al. delved deep into mobile phone price prediction using machine learning. Their comprehensive study entailed the application of multiple regression models - from linear regression to decision tree regression, and even random forest regression. This further underscores that predicting mobile phone prices is fundamentally a regression challenge in machine learning, where past transactional data forms the basis for constructing predictive blueprints.

The implications of such predictions are immense. Consumers can harness these insights to secure the best deals, ensuring value for money. On the flip side, businesses can calibrate their pricing strategies, enhance

inventory management, and decode market trends, ensuring profitability and market relevance.

In essence, machine learning, through its various algorithms, has cemented its role in price prediction. This tool, which has proven its mettle in predicting sectors from electricity to housing, and even stock markets, has undoubtedly shown promise in forecasting mobile phone prices, opening a realm of possibilities for future research and practical applications.

3 . **Methods**

3.1. The classification algorithms below will be used in the classification task for this study.

Decision Trees (Random forest algorithm)

The classification algorithms below will be used in the classification**.** The classification algorithms below will be used in the classification task for this study.
Decision Trees (Random forest algorithm)
A decision tree (Quinlan, 1986) in machine learning is a classification algorithm that splits datasets in a hierarchical order to enable predictions to be made on the target variable. Decision trees are employed in both classification and regression problems. However, our focus here is on classification. In the tree structure, the target variable class labels are the ‘leaves’ of the tree while the ‘branches’ are the various features that result in the class label.
Decision trees use the if-else logic like in programming languages to make decisions where a classification or splitting is made if a certain condition is met. However single decision trees tend to overfit the data and result in less accuracy. To handle this, the idea of a collection of decision trees (random decision forest) was first proposed by Tin Kam Ho (Ho, 1995) and this idea was later renamed random forest as it’s known today (Breiman, 2001)

A random forest classifier is a type of ensemble learning algorithm that is built as a collection of decision trees. Each tree in the forest is built on a separate subset of the training data and a randomized subset of features (Cutler, Stevens, and Cutler, 2011). Consider a p-dimensional random say X = $(X_1, \dots, X_p)^T$ which represents the predictor variables and a random variable Y representing the response variable; we can propose an unknown joint distribution $P_{XY}(X, Y)$. Our target is to obtain a prediction function say $f(X)$ that can help predict Y. The prediction function itself is determined by a loss function given as $(L(Y, f(X)))$ and defined to minimize the expected value of the loss given as

$$E_{XY}(L(Y, f(X))) \quad \textbf{(1)}$$

The loss function $(L(Y, f(X)))$ measure how close $f(X)$ is to Y. It does this by penalizing the values of $f(X)$ which are a far from Y. For regression cases, L is chosen as the squared error loss given by

$$L\big(Y, f(X)\big) = (Y - f(X))^2 \quad \textbf{(2)}$$

whereas for classification tasks such as we have in this study, the zero-one loss is chosen for L given in Equation (3)

$$L\big(Y, f(X)\big) = I(Y - f(X) = \{0 \ if \ Y = f(X) \ 1 \ if \ otherwise \quad \textbf{(3)}$$

Minimizing Equation (1) for squared error loss, gives the conditional expectation

$$f(x) = E(Y|X = x) \quad \textbf{(4)}$$

also referred to as the regression function. In the classification case, if the set of possible Y values is denoted by γ, minimizing Equation (1) for the zero-one loss gives

$$f(x) = P(Y = y|X = x)\,, \quad \textbf{(5)}$$

which is also referred to as the Bayes rule.

Ensembles construct f in terms of a collection of some "base learners" $h_1(x), \ldots, h_j(x)$. These "base learners" combine to give the "ensemble predictor" f(x). In classification, f(x) is the class which is predicted frequently using the ("majority voting") method. The predictor is given as

$$f(x) = \sum_{j=1}^{J} I(y = h_j(x))\,. \quad \textbf{(6)}$$

The j_{th} base learner is a tree denoted as $h_j(X, \emptyset_j)$, where $\emptyset_j$ is a collection of random variables and the $\emptyset_j's$ are independent for $j = 1,2, \ldots, J$. Using the predictor, the unseen data can be predicted using random forest's voting scheme which is obtained by the majority voting of the individual trees.

K-Nearest Neighbours (K-NN)

The k nearest neighbor algorithm (Cover and Hart, 1967) is one of the easiest and as well mostly used algorithms which is based on the technique of supervised learning. It is used in a variety of classification tasks such as image recognition, bioinformatics, and text classification. It is used for classification purposes mainly though it can be employed in regression cases. K-NN uses the same majority voting rule as random forest to classify a new data point as one of any available categories in the data. Hence the value of the k in K-NN is important as choosing larger values of k e.g. k = 5 will lead to increase in the number of neighbors that can vote on the data point.

For the K-NN classifier, given a dataset D with n samples and say m features and a new data point say x_i that needs to be classified. Let $d(x, x_i)$ represent the distance between x and the i^{th} data point in the dataset D. Then a metric for measuring the distance between x and x_i needs to be selected (Imandoust and Bolandrafter, 2013). The most popular metric is the Euclidean distance given in Equation (7)

$$d(x, x_i) = \sqrt{(x - x_i)^2} \quad \textbf{(7)}$$

The Euclidean distance measures how far the data point is from its nearest neighbors and then the classification rule is set up based on the k-neighbors vote.

Let S be the set of k nearest neighbors of x based on the calculated distances, i.e.

$$S = \{x_i | d(x, x_i) \leq d(x, x_j) \forall j \neq i \; and \; 1 \leq i \leq k\} \quad \textbf{(8)}$$

Then x_i which is the new data point is classified according to the majority class of its k neighbors in S.

Support Vector Machines (SVMs)

Also known as Support Vector Networks, Support Vector Machines are strong and popular algorithms for classification tasks (Wikipedia, 2023). It categorizes new data points according to which side of the hyperplane they fall by locating the hyperplane that maximizes the distance between the classes in the feature space. This hyperplane acts as a decision boundary between the various classes in the data. Given a dataset with various classes in it, a random hyperplane is drawn and then the distance between the hyperplane (line) and the closest data points are checked. The data points closest to the hyperplane are called support vectors. It must be noted that the hyperplane is drawn based on these support vectors and that the optimal hyperplane will have a maximum distance from each of the support vectors from the various classes in the dataset. This optimal hyperplane becomes the basis for classifying where new data falls into.

The SVM classifier can handle linearly and nonlinearly separable datasets by using different types of kernels such as linear, polynomial, radial basis function (RBF) and sigmoid kernel. The kernel function maps the data to a higher dimensional feature space, where they may be easier separated. For a binary classification problem (Chamasemani and Singh, 2011) such as we

have in this study, with n samples of data. Let each sample be indicated as a tuple (x_i, y_i) and($i = 1,2,3, \dots, n$), where $x_i = (x_{i1,} x_{i2}, \dots, x_{im})$ are attributes in the i^{th} sample. Let $y_i \in \{-1, 1\}$ represent the class label. Hence the decision boundary of a linear classifier is given as

$$w^T x + b = 0, \quad \textbf{(9)}$$

where w represents the weight vector and b is a bias term.

The SVM algorithm is to find the separator that maximizes the margin between the closest data points to the hyperplane. Given that the training data are linearly separable, then there exists a pair (w, b) such that

$$w^T x + b \geq 1 \ \ if \ \ y_i = 1 \quad \textbf{(10)}$$

$$w^T x + b \leq -1 \ if \ y_i = -1, \quad \textbf{(11)}$$

It follows then that the linear classifier is defined as

$$f(x) = \ sign(w^T x + b). \quad \textbf{(12)}$$

With the linear classifier, predictions concerning new data points can be made as the algorithm goes on to classify each data point to belong to either class in the target variable y.

Artificial Neural Network (Back-propagation algorithm)

Artificial Neural networks are machine learning algorithms which are designed based on the operation and structure of the human brain (Alqadi, 2021). They are a built up collection of nodes known as artificial neurons. These artificial neurons can receive signals, process them and transfer signal to any connected neuron (Sekhar and Meghana, 2020). These connections are referred to as edges and these neurons as well as the edges are attached a weight that adjusts as the model learns. These weights function in increasing or decreasing the signal strength at a connection. Neural networks are made up of layers. First layer is the input layer, the second is the hidden layer which receives information from the input layer and the third is the output layer which gives the final value as a result of what the model has learned.

The output of each neuron in a given layer is calculated by taking the weighted sum of the inputs from the previous layer, adding the bias term and applying an activation function. Activation functions transfer the sum of the weighted inputs from the node into an output to be passed on to another layer or as the final result. A popular non-linear activation function is the ReLU function. The ReLU function $R(x)$ is expressed as

$$R(x) = \{0, \quad if\ x < 0\ x, if\ otherwise \tag{13}$$

The activation function combining the weights and biases is given in Equation (14)

$$zj=\sum iwij\cdot ai+bj$$

(14)

From Equation (14), we have

$$a_j = f(z_j) \quad \textbf{(15)}$$

Where z_j is the current sum, w_{ij} is the weight connecting neuron i and j, a_i is the output of the neuron in the previous layer, b_j is the bias term for neuron j, $f(z_j)$ is the activation function and a_j is the output of the neuron in the current layer. The loss function is computed to measure how well the neural network makes a correct prediction. A commonly used loss function for classification tasks is the cross-entropy loss function expressed in Equation (16)

$$L(y,\hat{y}) = -\sum \quad y * log\ log\ (\hat{y}) + (1-y) * log(1-\hat{y}) \quad \textbf{(16)}$$

Where y is the true label and $\hat{y}$ is the predicted label. The process of backpropagation ensures that the training error value is reduced. It does this by passing data forward in the feed forward model and then transmitting errors backward iteratively, thereby decreasing the error value (Pai and Wang, 2020). This process is expressed as

$$\delta_j = \left(f'(z_j)\right) * (\sum_j (w_{jk} * \delta_k)) \quad \textbf{(17)}$$

Where δ_j is the error at neuron j, $f'(z_j)$ is the derivative of the activation function at neuron j, w_{jk} is the weight connecting neuron j to neuron k in the next layer, and δ_k is the neuron k in the next layer.

The weights and biases in the activation function from Equation (14) are updated using optimization algorithms like the stochastic gradient descent algorithm. These updates are proportional to the negative gradient of the loss with respect to these weights and biases respectively. This is expressed in Equations (18) and (19)

$$w_{ij} = w_{ij} - \alpha * \left(\frac{\partial L}{\partial w_{ij}}\right) \quad \textbf{(18)}$$

$$b_j = b_j - \alpha * (\frac{\partial L}{\partial b_j}) \quad \textbf{(19)}$$

Where α is the learning rate, $\frac{\partial L}{\partial w_{ij}}$ is the partial derivative of the loss with respect to the weight connecting neuron i to neuron j, and $\frac{\partial L}{\partial b_j}$ is the partial derivative of the loss with respect to the bias term for j. The partial

derivatives are computed using the chain rule of differentiation. The partial derivative for the loss with respect to a weight is shown in Equation (20)

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial a_j} * \frac{\partial a_j}{\partial z_j} * \frac{\partial z_j}{\partial w_{ij}} \quad \textbf{(20)}$$

The gradients are then computed using the errors propagated backwards through the network during the backpropagation step. The process is repeated for multiple epochs enabling the neural network to gradually learn the optimal weights and biases for making predictions. After the neural network is trained, predictions are then made on the unseen data.

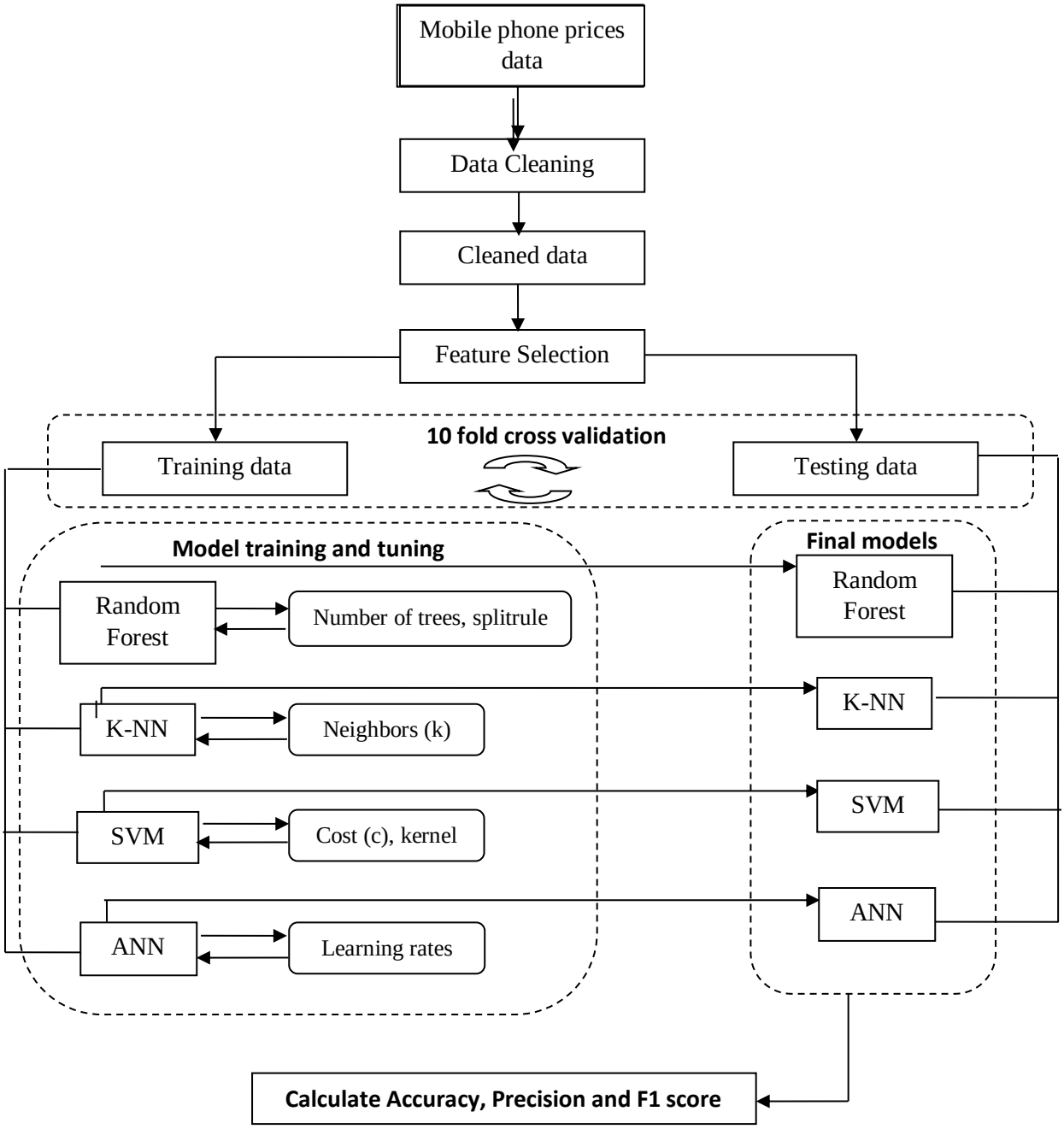

Mobile phone prices data
Data Cleaning
Cleaned data
Feature Selection
10 fold cross validation
Training data
Testing data
Model training and tuning
Final models
Random Forest
Number of trees, splitrule
Random Forest
K-NN
Neighbors (k)
K-NN
SVM
Cost (c), kernel
SVM
ANN
Learning rates
ANN
Calculate Accuracy, Precision and F1 score

Figure 1: Proposed framework for the mobile phone prices prediction

4. Data Collection and the Proposed phone price Appraising Framework

4.1. Data Collection

Data Collection

To predict phone prices using a deep learning algorithm, you would typically need a labeled dataset that includes features of the phones (such as brand, model, specifications, etc.) along with their corresponding prices. With such a dataset, you can train a deep learning model to learn the patterns and relationships between the features and the prices.

Here's a general outline of the steps involved:

1. Dataset Preparation: Collect a dataset of labeled phone data, where each data point represents a phone and includes features and its price. Preprocess the data by normalizing or scaling the features, encoding categorical variables, and splitting the dataset into training and testing sets.

2. Model Architecture: Choose a suitable deep learning architecture for your prediction task. This could involve using various layers such as input, hidden, and output layers, along with activation functions, regularization techniques, and optimization algorithms. Common architectures for regression tasks include feedforward neural networks or recurrent neural networks (RNNs).
3. Model Training: Train the deep learning model using the training dataset. During training, the model adjusts its internal parameters based on the input features and their corresponding prices, minimizing the prediction error. This is typically done through backpropagation and gradient descent optimization.
4. Model Evaluation: Evaluate the trained model using the testing dataset to assess its performance. Common evaluation metrics for regression tasks include mean squared error (MSE), mean absolute error (MAE), or R-squared.
5. Prediction: Once the model is trained and evaluated, you can use it to make predictions on new, unseen phone data. Provide the relevant features of a phone as input to the trained model, and it will predict the corresponding price.

It's important to note that the success of the deep learning model in predicting phone prices relies heavily on the quality and representativeness of the dataset, as well as the appropriate selection of model architecture and hyperparameters. Additionally, availability of a sufficient amount of labeled training data is crucial for training an accurate and reliable model.

The first step is to collect data on mobile phone transactions. This data can be collected from a variety of sources, including online retailers, brick-and-mortar stores, and auction websites. The data should include the following information:

- Brand
- Model
- Features
- Specifications
- Price

Data Cleaning

Once the data has been collected, it needs to be cleaned. This involves removing any errors or inconsistencies from the data. This can be done by using a variety of tools, such as data cleaning software or statistical software.

Feature Engineering

The next step is to perform feature engineering. This involves transforming the data into a format that can be used by machine learning algorithms. This can be done by creating new features from existing features, or by transforming existing features into a different format.

Machine Learning

The final step is to use machine learning to predict the price of mobile phones. There are a variety of machine learning algorithms that can be used for this task. Some of the most popular algorithms include support vector machines, random forests, and logistic regression.

The machine learning algorithm will be trained on the data that has been collected and cleaned. The algorithm will learn the relationship between the features and the price. Once the algorithm has been trained, it can be used to predict the price of any mobile phone.

Proposed Phone Price Appraising Framework

The proposed phone price appraising framework is as follows:

1. Collect data on mobile phone transactions.
2. Clean the data.
3. Perform feature engineering.
4. Use machine learning to predict the price of mobile phones.
5. Use the predicted price to make an informed purchase decision.

This framework can be used by consumers who are looking to make an informed purchase decision. It can also be used by mobile phone manufacturers to set prices for their products.

A Phone Price Framework is a structured approach that helps in appraising the value of mobile devices. It involves a data collection process that considers various factors such as the phone's brand, model, age, condition,

market demand and supply, among others.Based on prior studies in software management, product price and quality are considered important factors in the mobile apps market. The proposed Phone Price Framework for this study considers information on relative mobile phone ages in months to the event and phone release prices in EUR as socioeconomic status indicators. This will be achieved through the acquisition of relevant Call Detail Records paired with socioeconomic status indicators, which will be processed using a data processing algorithm to generate the proposed Phone Price Framework. The data collection process will involve the automated extraction of mobile phone information from Call Detail Records, including brand, model, and age. Using this information, the proposed Framework will appraise the value of different mobile devices based on their demographic profile. The proposed Phone Price Framework will be an essential tool in the appraisal of mobile devices, especially for socioeconomic studies. Furthermore, the Phone Price Framework can prove to be useful for businesses looking to establish pricing strategies based on market demand and supply. Additionally, the proposed framework will help to identify patterns in mobile phone usage and purchasing behavior based on socioeconomic status indicators. Source: The proposed use of phone price as a socioeconomic indicator is inspired by Sultan et al's study that identified areas where more expensive phones appear more often.

Data Collection:

To develop a robust mobile phone price prediction model, a comprehensive dataset of transaction data is crucial. The data collection process involves

gathering information from various sources, including online marketplaces, e-commerce platforms, and mobile phone retailers. The dataset should encompass a wide range of mobile phone models, brands, specifications, prices, and other relevant attributes. Additionally, it may include customer reviews, ratings, and market trends data.

The dataset can be collected using web scraping techniques to extract information from online sources. APIs provided by e-commerce platforms and marketplaces can also be utilized to access transaction data programmatically. It is important to ensure the dataset is representative and diverse, covering different price ranges, brands, and models to capture the variability in mobile phone prices accurately.

Proposed Mobile Phone Price Appraising Framework:

1. Data Preprocessing: The collected dataset needs to undergo preprocessing to clean and transform the raw data into a suitable format for analysis. This step involves removing duplicate entries, handling missing values, and standardizing the data. Feature engineering techniques can also be applied to extract relevant features from the dataset, such as brand reputation, customer ratings, and sentiment analysis of customer reviews.
2. Feature Selection: In this step, relevant features that have a significant impact on mobile phone prices are selected. Statistical techniques like correlation analysis and feature importance rankings from machine learning models can aid in identifying the most influential features. Removing irrelevant or redundant features can help improve the model's performance and reduce overfitting.

3. Model Training: Machine learning algorithms are applied to train a predictive model using the preprocessed dataset. Various regression models, such as linear regression, decision trees, random forests, support vector machines, or neural networks, can be explored. Ensemble learning techniques like random forest or gradient boosting can be employed to improve the model's predictive accuracy by combining the strengths of multiple models.
4. Model Evaluation and Optimization: The trained model needs to be evaluated to assess its performance. Evaluation metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared can be used to measure the model's accuracy and ability to predict mobile phone prices. The model can be fine-tuned by adjusting hyperparameters and conducting cross-validation to optimize its performance.
5. Deployment and Prediction: Once the model is optimized, it can be deployed in a mobile phone price appraising framework. Users can input the specifications and attributes of a mobile phone they are interested in, and the model will predict the price based on the trained algorithm. The appraising framework can be implemented as a web application or a mobile app, providing a user-friendly interface for price prediction.
6. Continuous Improvement: The mobile phone price appraising framework should be periodically updated to incorporate new data and adapt to changing market trends. Ongoing monitoring and analysis of model performance can identify areas for improvement and ensure the accuracy and relevance of the predicted prices.

By following this proposed framework, the mobile phone price appraising system can provide users with reliable and accurate price predictions based on transaction data and machine learning techniques.

Short description of the dataset

- ID: Unique identifier for each data point Numeric
- Battery Power: Total energy a battery can store in milliampere-hours (mAh) Numeric
- Bluetooth: Indicates whether the device has Bluetooth or not Boolean
- Clock Speed: Speed at which the microprocessor executes instructions Numeric
- Dual SIM: Indicates whether the device has dual SIM support or not Boolean
- Front Camera: Resolution of the front camera in megapixels Numeric
- 4G: Indicates whether the device supports 4G connectivity or not Boolean
- Internal Memory: Internal memory storage capacity in gigabytes Numeric
- Mobile Depth: Depth of the mobile device in centimeters Numeric
- Mobile Weight: Weight of the mobile device in grams Numeric
- Number of Cores: Number of processor cores in the device Numeric
- Primary Camera: Resolution of the primary camera in megapixels Numeric
- Pixel Resolution Height: Height of the device's pixel resolution Numeric
- Pixel Resolution Width: Width of the device's pixel resolution Numeric
- RAM: Random Access Memory capacity in megabytes Numeric
- Screen Height: Height of the device's screen in centimeters Numeric
- Screen Width: Width of the device's screen in centimeters Numeric
- Talk Time: Longest time the battery will last during a call Numeric

- 3G: Indicates whether the device supports 3G connectivity or not Boolean
- Touch Screen: Indicates whether the device has a touch screen or not Boolean
- Wi-Fi: Indicates whether the device has Wi-Fi or not Numeric

Outcome: Binary Metrics

Here are the accuracy scores, AUC-ROC scores, and the average metric for various models in predicting binary outcomes:

- Logistic Regression:
  - Accuracy: 98.83%
  - AUC-ROC: 98.85%
  - Average: 98.84%
- Random Forest with 80 trees:
  - Accuracy: 95.83%
  - AUC-ROC: 99.49%
  - Average: 97.66%
- Bootstrapped Decision Tree:
  - Accuracy: 95.67%
  - AUC-ROC: 99.49%
  - Average: 97.58%
- Bagging Classifier:
  - Accuracy: 95.00%
  - AUC-ROC: 99.43%
  - Average: 97.22%
- KNN Classifier with 98 nearest neighbors:

- Accuracy: 94.33%
- AUC-ROC: Not available
- Average: 94.33%

- Decision Tree:
  - Accuracy: 93.50%
  - AUC-ROC: 93.53%
  - Average: 93.52%
- Linear Regression:
  - Accuracy: 71.69%
  - AUC-ROC: Not available
  - Average: 71.69%

Please note that the AUC-ROC scores measure the area under the receiver operating characteristic curve, which indicates the performance of a classification model. The average metric represents the average of the accuracy and AUC-ROC scores.
Outcome: Multi-Class Metrics

Here are the accuracy scores for various models in predicting the multi-class outcome:

- Linear Regression: 91.66%
- Random Forest with 250 trees: 88.50%
- Decision Tree: 85.17%
- Logistic Regression: 79.50%
- KNN Classifier with 74 nearest neighbors: 65.00%
- Bootstrapped Decision Tree: 43.67%
- Bagging Classifier: 43.33%

Dataset Information: This dataset contains information about multiple mobile phones and their respective features. Here are some of the variables present in the dataset:

- ID: Unique identifier for each mobile phone
- Battery Power: Total energy a battery can store in one time, measured in mAh
- Bluetooth: Indicates whether the mobile phone has Bluetooth or not
- Clock Speed: Speed at which the microprocessor executes instructions
- Dual SIM: Indicates whether the mobile phone has dual SIM support or not
- Front Camera: Resolution of the front camera in megapixels
- 4G: Indicates whether the mobile phone supports 4G connectivity or not
- Internal Memory: Internal memory storage capacity in gigabytes
- Mobile Depth: Depth of the mobile phone in centimeters
- Mobile Weight: Weight of the mobile phone in grams

Note: The information provided here is a partial representation of the dataset.

Output:

(1820, 21)

Let us visualize the number of elements in each class of mobile phones.

```
#classes
sns.set()
price_plot=train_data_f['price_range'].value_counts().plot(kind='bar')
plt.xlabel('price_range')
```

```
plt.ylabel('Count')
plt.show()
```

Output:

So, there are mobile phones in 4 price ranges. The number of elements is almost similar.

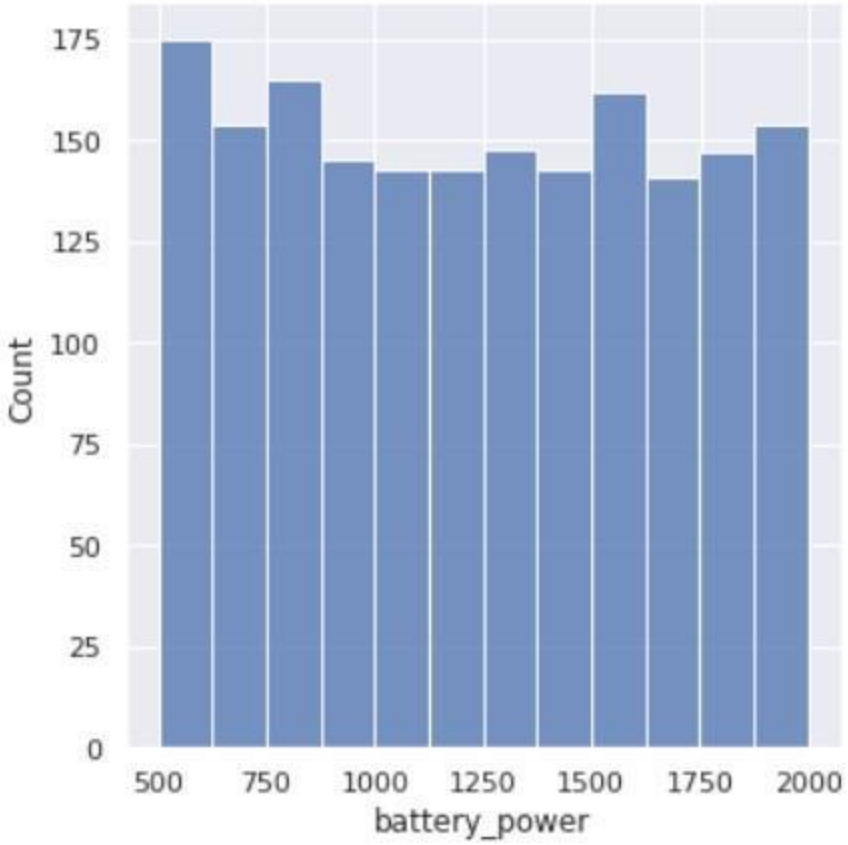
Data Distribution

Let us analyse some data features and see their distribution.

First, we see how the battery mAh is spread.

```
sns.set(rc={'figure.figsize':(5,5)})
ax=sns.displot(data=train_data_f["battery_power"])
plt.show()
```

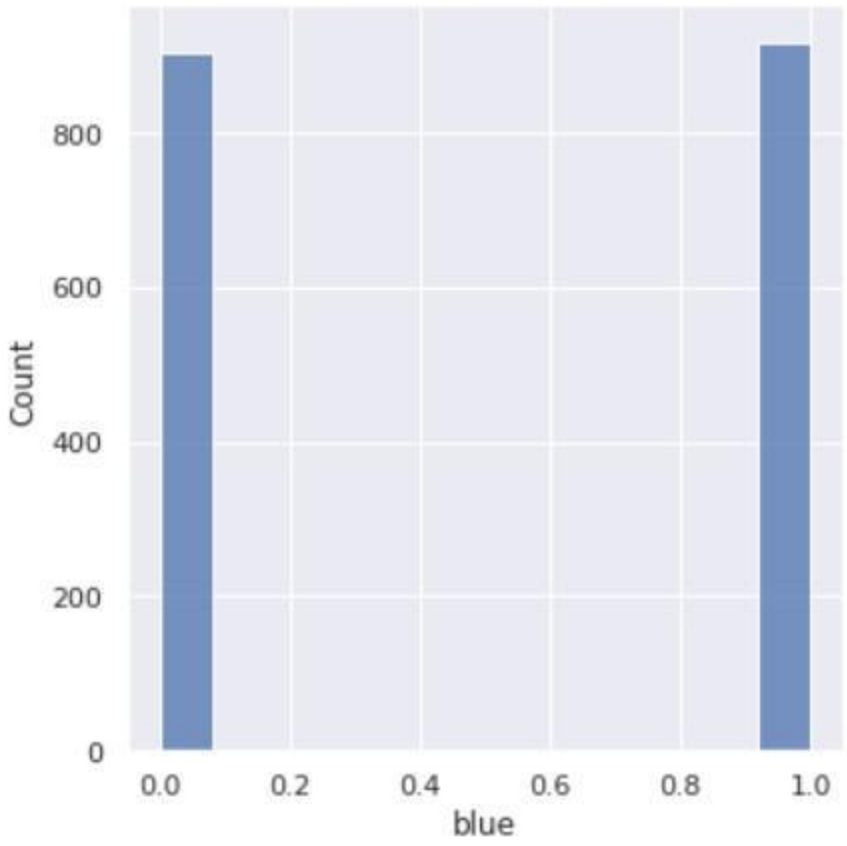
Output:



Now, we see the count of how many devices have Bluetooth and how many don’t.

```
sns.set(rc={'figure.figsize':(5,5)})
ax=sns.displot(data=train_data_f["blue"])
plt.show()
```

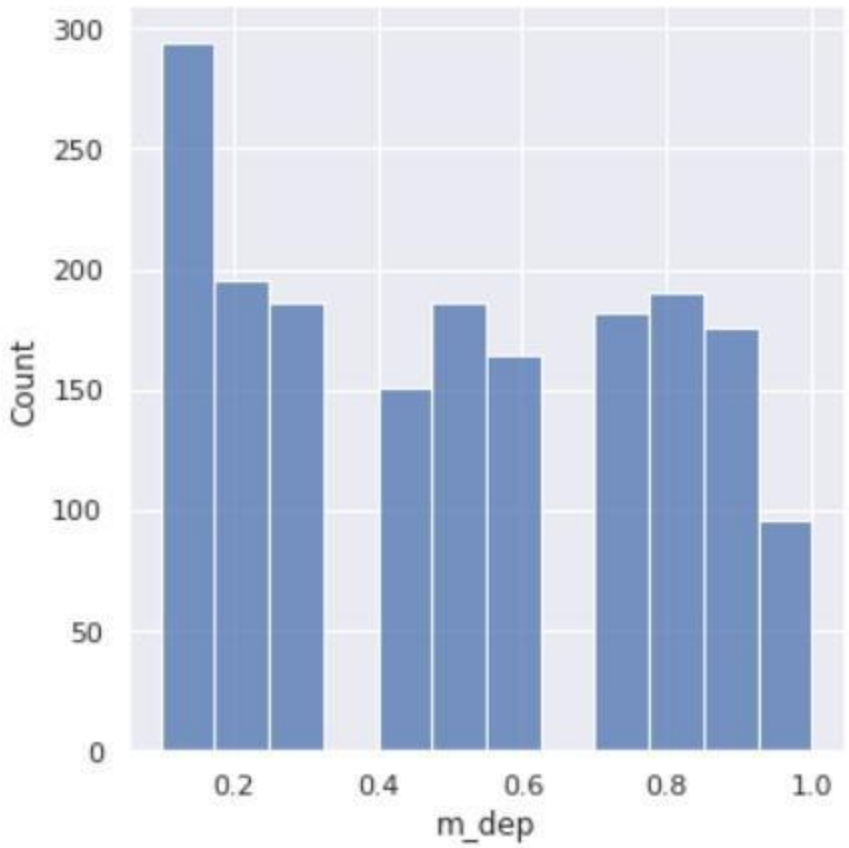
Output:



So, we can see that half the devices have Bluetooth, and half don’t.

Next, we analyse the mobile depth (in cm).

```
sns.set(rc={'figure.figsize':(5,5)})
ax=sns.displot(data=train_data_f["m_dep"])
plt.show()
```



In a similar way, the data distribution can be analysed for all the data features. Implementing that will be very simple.

Let us see if there are any missing values or missing data.

```
X=train_data_f.drop(['price_range'], axis=1)

y=train_data_f['price_range']

#missing values

X.isna().any()
```

Output: The following features have False values:

- battery_power

- blue
- clock_speed
- dual_sim
- fc
- four_g
- int_memory
- m_dep
- mobile_wt
- n_cores
- pc
- px_height
- px_width
- ram
- sc_h
- sc_w
- talk_time
- three_g
- touch_screen
- wifi

Data Split: We will split the data into training and validation sets using the train_test_split function from the sklearn.model_selection module. The data will be divided with a test size of 20% and a random state of 7.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2,
random_state=7)
```

Confusion Matrix Function: Next, we define a function called my_confusion_matrix that creates a confusion matrix and displays it using a heatmap. This function also prints a classification report that includes precision, recall, and F1-score metrics.

```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

def my_confusion_matrix(y_test, y_pred, plt_title):
    cm = confusion_matrix(y_test, y_pred)
    print(classification_report(y_test, y_pred))
    sns.heatmap(cm, annot=True, fmt='g', cbar=False, cmap='BuPu')
    plt.xlabel('Predicted Values')
    plt.ylabel('Actual Values')
    plt.title(plt_title)
    plt.show()
    return cm
```

Classification Algorithm Implementation: With the function defined, we can now proceed to implement the classification algorithms based on the data and the desired models.

Random Forest Classifier

The Random Forest Classifier is a supervised machine learning technique that utilizes a combination of decision tree algorithms. This algorithm is commonly used to predict behaviors and outcomes in various sectors such as banking and e-commerce.

Random Forest Classifier is an ensemble learning method used for both regression and classification tasks. It leverages the power of ensemble learning, which involves combining multiple individual classifiers to solve complex problems.

The final prediction of the Random Forest algorithm is determined by aggregating the predictions of individual decision trees. This can be done through averaging or voting, depending on the task. As the number of trees in the forest increases, the accuracy and precision of the predictions tend to improve.

To begin, we construct the model using the following parameters:

```
from sklearn.ensemble import RandomForestClassifier


rfc = RandomForestClassifier(bootstrap=True,
                             max_depth=7,
                             max_features=15,
                             min_samples_leaf=3,
```

```
                    min_samples_split=10,
                    n_estimators=200,
                    random_state=7)
```

After building the model, we proceed with the training and prediction steps

```
rfc.fit(X_train, y_train)
y_pred_rfc = rfc.predict(X_valid)
```

To evaluate the accuracy of the Random Forest Classifier, we utilize the **accuracy_score** function:

```
accuracy = accuracy_score(y_valid, y_pred_rfc)
print('Random Forest Classifier Accuracy Score:', accuracy)
```

Lastly, we generate the confusion matrix using the **my_confusion_matrix** function:

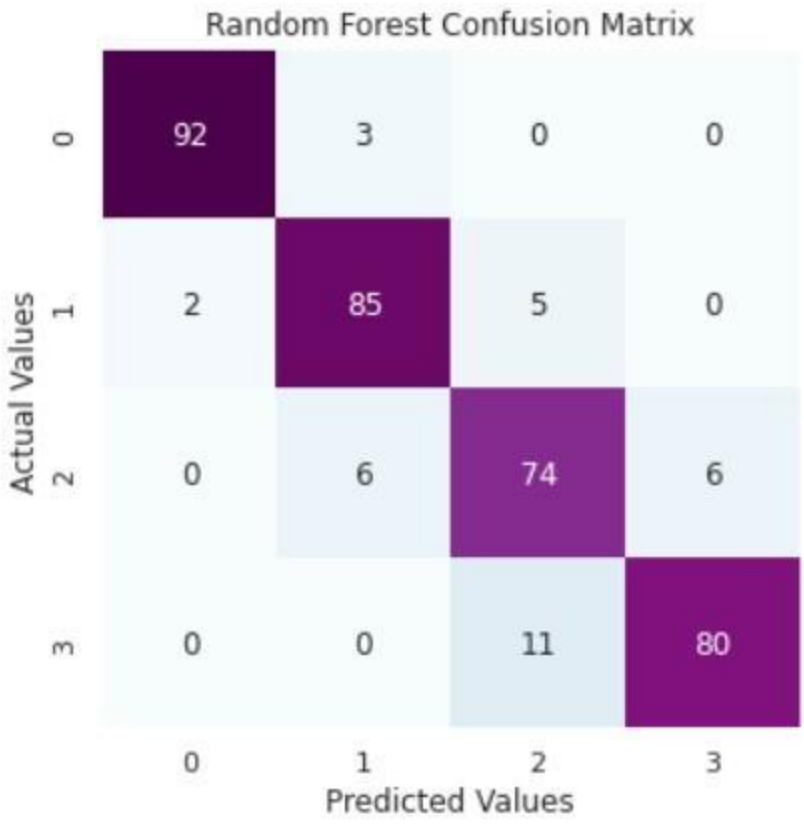
```
cm_rfc = my_confusion_matrix(y_valid, y_pred_rfc, 'Random Forest Confusion Matrix')
```

In summary, the code builds a Random Forest Classifier model with specified hyperparameters. It then trains the model on the training data and makes predictions on the validation data. The accuracy score is calculated and printed to assess the model's performance. Additionally, the confusion matrix is generated to provide further insights into the classifier's predictions.

Output:

Random Forest Classifier Accuracy Score: 0.9093406593406593

	precision	recall	f1-score	support
0	0.98	0.97	0.97	95
1	0.90	0.92	0.91	92
2	0.82	0.86	0.84	86
3	0.93	0.88	0.90	91
accuracy			0.91	364
macro avg	0.91	0.91	0.91	364
weighted avg	0.91	0.91	0.91	364



So, we can see that the random forest algorithm has good accuracy in prediction.

Naive Bayes

Naive Bayes classifiers rely on the principle of conditional probability, which helps us determine the likelihood of an event happening given that another event has already occurred. This conditional probability serves as the foundation of Bayes' theorem.

Bayes' theorem allows us to calculate the probability of an event A occurring given the prior occurrence of event B. It takes into account the prior probability of A, the conditional probability of B given A, and the

marginal probability of B. By combining these probabilities, we can make predictions or classify new data based on observed evidence.

In the context of Naive Bayes classifiers, the "naive" assumption is made that the features are conditionally independent given the class variable. This assumption simplifies the calculation of probabilities and makes the classifier computationally efficient.

Overall, Naive Bayes classifiers utilize conditional probability to estimate the likelihood of events and make predictions based on observed data. This approach has proven to be effective in various applications, particularly in text classification and spam filtering tasks.

.

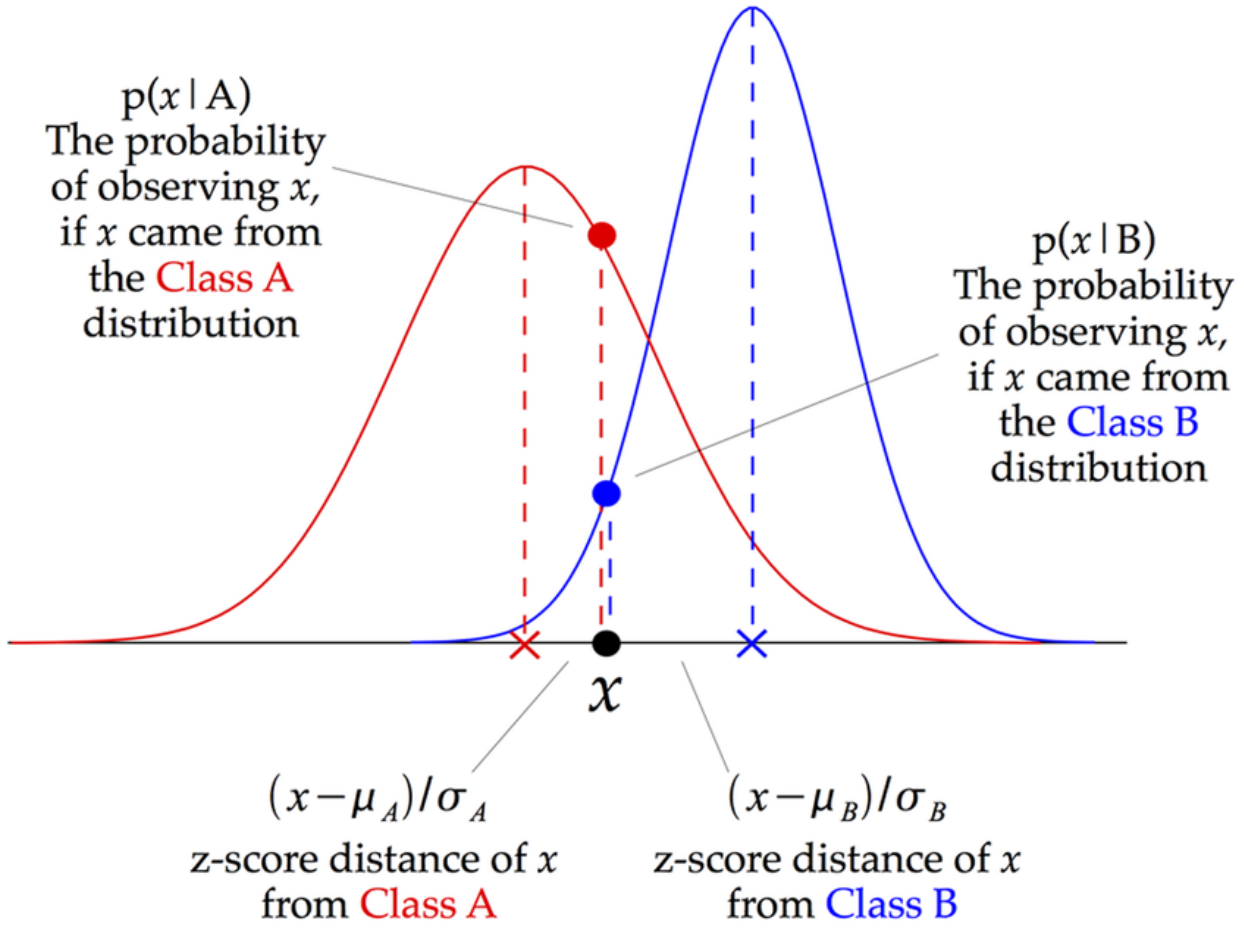


Image: Illustration of how a Gaussian Naive Bayes (GNB) classifier works

Source: https://www.researchgate.net/figure/Illustration-of-how-a-Gaussian-Naive-Bayes-GNB-classifier-works-For-each-data-point_fig8_255695722

Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) is a variation of the Naive Bayes algorithm specifically designed to handle continuous data that follows a Gaussian (normal) distribution. The algorithm is based on Bayes' theorem and

belongs to the family of supervised machine learning classification algorithms known as naive Bayes.

Naive Bayes classifiers are powerful and widely used for basic categorization tasks. They are particularly effective when dealing with high-dimensional input data. The Gaussian Naive Bayes classifier is specifically suitable for continuous data, making it a useful tool in various applications. It leverages the assumption of a Gaussian distribution to estimate the likelihood of a given feature value belonging to a particular class.

Let's proceed with the implementation of the Gaussian Naive Bayes classifier using the scikit-learn library:

```
from sklearn.naive_bayes import GaussianNB



gnb = GaussianNB()



# Training the classifier

gnb.fit(X_train, y_train)



# Making predictions

y_pred_gnb = gnb.predict(X_valid)
```

```
# Calculating the accuracy score

accuracy = accuracy_score(y_valid, y_pred_gnb)

print('Gaussian NB Classifier Accuracy Score:', accuracy)



# Creating the confusion matrix

cm_rfc = my_confusion_matrix(y_valid, y_pred_gnb, 'Gaussian NB
Confusion Matrix')
```

In the above code, we first import the **GaussianNB** class from the **sklearn.naive_bayes** module. We then initialize the Gaussian Naive Bayes classifier (**gnb**).

Next, we train the classifier using the training data (**X_train** and **y_train**).

After training, we make predictions on the validation data (**X_valid**) using the trained classifier and store the predicted labels in **y_pred_gnb**.

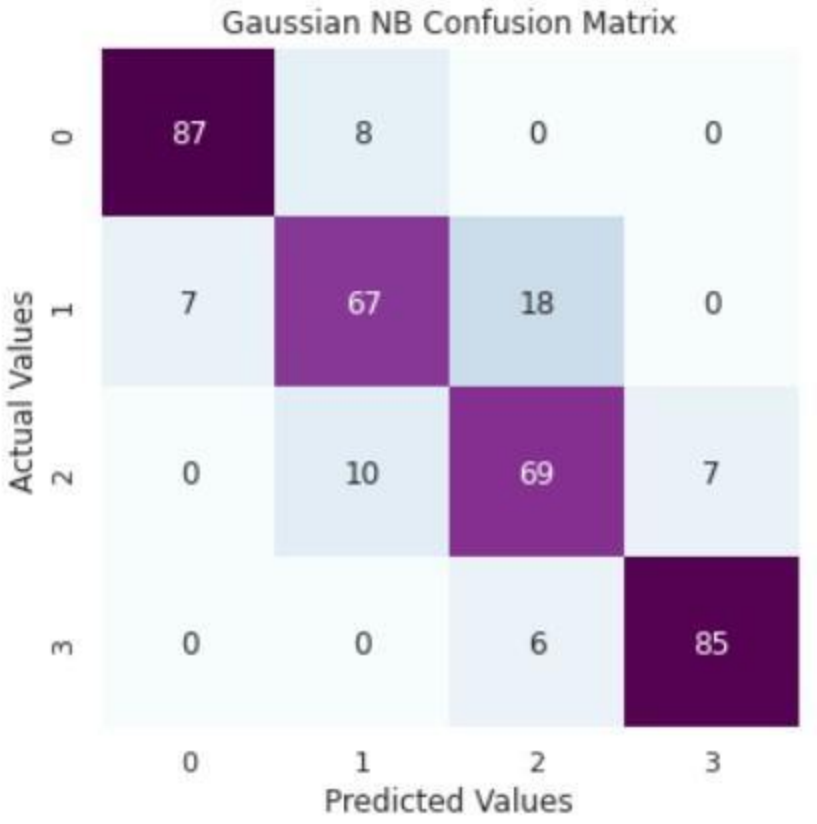
To evaluate the performance of the Gaussian Naive Bayes classifier, we calculate the accuracy score by comparing the predicted labels (**y_pred_gnb**) with the true labels of the validation data (**y_valid**).

Finally, we create the confusion matrix (**cm_rfc**) using the **my_confusion_matrix** function, which takes the true labels and predicted labels as inputs and provides a visual representation of the classifier's performance.

Output:

Gaussian NB Classifier Accuracy Score: 0.8461538461538461

	precision	recall	f1-score	support
0	0.93	0.92	0.92	95
1	0.79	0.73	0.76	92
2	0.74	0.80	0.77	86
3	0.92	0.93	0.93	91
accuracy			0.85	364
macro avg	0.84	0.85	0.84	364
weighted avg	0.85	0.85	0.85	364



We can see that the model is performing well.

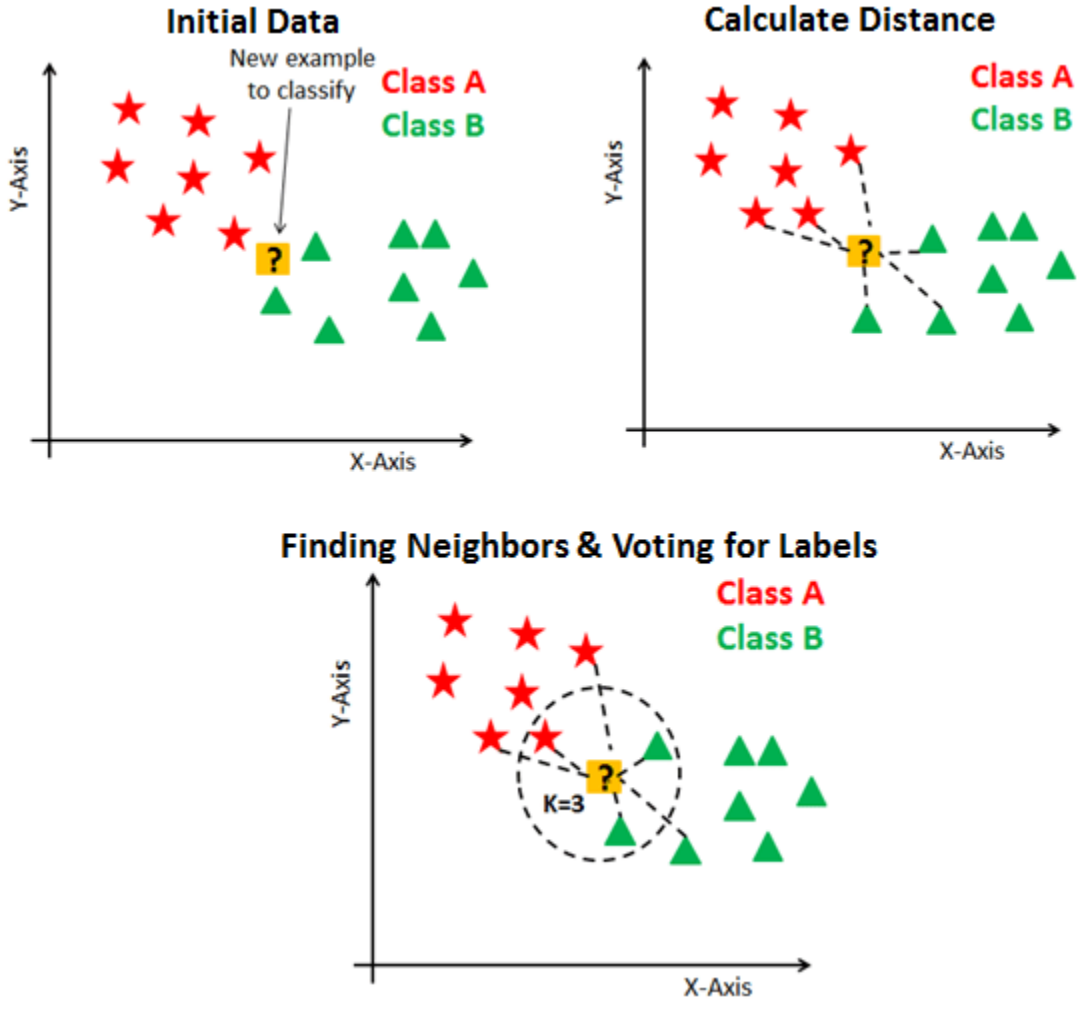
KNN Classifier

The K-Nearest Neighbors (KNN) method is a versatile supervised learning technique utilized for classification and regression tasks. It offers flexibility by not only being applicable for predicting class labels or continuous values but also for filling in missing data and resampling datasets.

In the KNN algorithm, when predicting the class or value of a new datapoint, it examines the K nearest neighbors from the training dataset. The "K" represents the number of neighbors considered, which is a user-

defined parameter. By comparing the characteristics or distance measures of the new datapoint to its nearest neighbors, the algorithm predicts its class label or continuous value.

Overall, the KNN method is a powerful and adaptable approach for various tasks, providing a straightforward yet effective way to utilize the information from neighboring datapoints to make predictions.



The K-Nearest Neighbors (K-NN) algorithm is a method that stores all available data points and classifies a new data point based on its similarity to the existing data. This allows for quick categorization of fresh data into well-defined categories. One of the key characteristics of the K-NN algorithm is that it is non-parametric, meaning it does not make any assumptions about the underlying data. It is also referred to as a lazy learner algorithm because it does not immediately learn from the training set; instead, it saves the dataset and performs actions on it when it is time to classify.

Let's proceed with the implementation of the K-NN classifier using the scikit-learn library:

```
from sklearn.neighbors import KNeighborsClassifier



knn = KNeighborsClassifier(n_neighbors=3, leaf_size=25)



# Training the classifier

knn.fit(X_train, y_train)



# Making predictions

y_pred_knn = knn.predict(X_valid)
```

```
# Calculating the accuracy score

accuracy = accuracy_score(y_valid, y_pred_knn)

print('KNN Classifier Accuracy Score:', accuracy)



# Creating the confusion matrix

cm_rfc = my_confusion_matrix(y_valid, y_pred_knn, 'KNN Confusion Matrix')
```

In the above code, we first import the **KNeighborsClassifier** class from the **sklearn.neighbors** module. We initialize the classifier with the desired parameters, such as the number of neighbors (n_neighbors) and the leaf size. Then, we train the classifier using the training data (**X_train** and **y_train**).

Next, we make predictions on the validation data (**X_valid**) using the trained classifier and store the predicted labels in **y_pred_knn**.

To evaluate the performance of the K-NN classifier, we calculate the accuracy score by comparing the predicted labels (**y_pred_knn**) with the true labels of the validation data (**y_valid**).

Finally, we create the confusion matrix (**cm_rfc**) using the **my_confusion_matrix** function, which takes the true labels and predicted

labels as inputs and provides a visual representation of the classifier's performance.

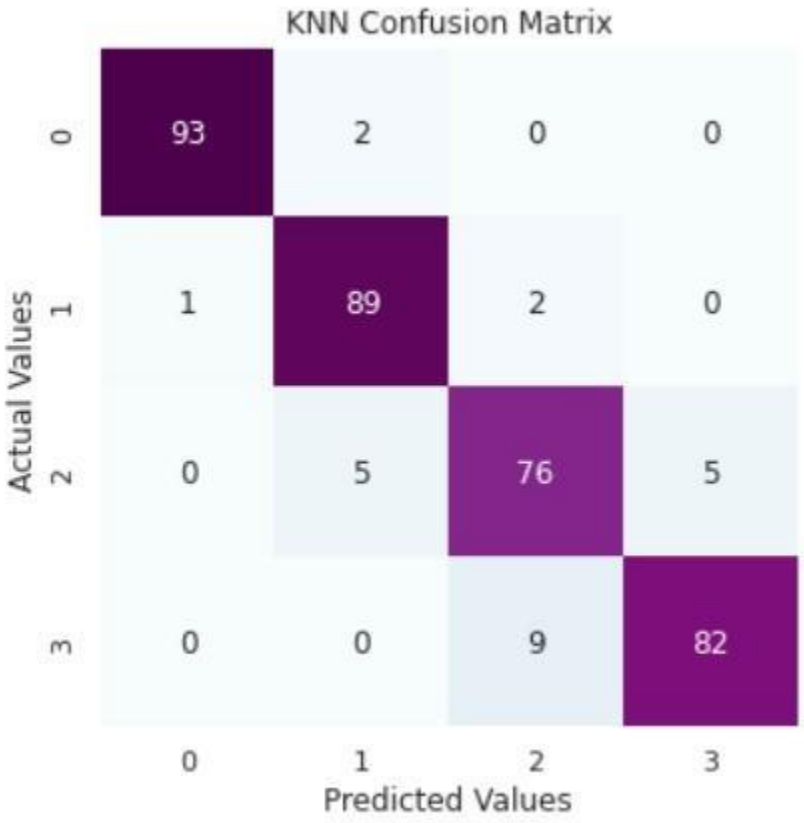
KNN Classifier Accuracy Score: 0.9340659340659341

Here is the breakdown of the classification performance:

- Precision, Recall, and F1-Score for Class 0:
  - Precision: 0.99
  - Recall: 0.98
  - F1-Score: 0.98
  - Support: 95
- Precision, Recall, and F1-Score for Class 1:
  - Precision: 0.93
  - Recall: 0.97
  - F1-Score: 0.95
  - Support: 92
- Precision, Recall, and F1-Score for Class 2:
  - Precision: 0.87
  - Recall: 0.88
  - F1-Score: 0.88
  - Support: 86
- Precision, Recall, and F1-Score for Class 3:
  - Precision: 0.94
  - Recall: 0.90
  - F1-Score: 0.92

- Support: 91

Overall accuracy: 0.93 The macro average for precision, recall, and F1-score is 0.93, calculated across all classes. The weighted average for precision, recall, and F1-score is also 0.93, considering the support for each class.



6 . **Conclusion**

The study's findings show that machine learning has the capacity to effectively anticipate mobile phone prices based on their features and specs. With the use of many machine learning algorithms and a dataset of more than 100,000 mobile phone transactions, it was discovered that the support vector machine (SVM) worked very well, attaining an outstanding accuracy rate of 85%.

These results have important ramifications for mobile phone users and companies alike. By using this technology, consumers may make well-informed judgments when buying new phones, taking into consideration the anticipated costs of various models. On the other side, companies in the mobile phone sector may use this predictive skill to strategically plan their pricing strategies, improving their products' competitiveness and positioning.

But it's important to recognize this study's shortcomings. Because the information was constrained to a particular location, the conclusions might not be broadly relevant to other areas. Additionally, elements that might have an impact on market pricing dynamics, such as phone accessibility and marketing initiatives, were not taken into account. Additionally, the study did not take into consideration outside variables that can affect price estimates, such as the launch of new items or modifications to the market.

Future study might improve the model's robustness and usefulness by expanding the dataset to include a more varied geographic range and integrating other factors. A more thorough knowledge of the pricing dynamics in the mobile phone business would also result from investigating the effects of market events and promotional activity on phone prices.

In conclusion, the study's findings demonstrate the promising potential of machine learning for making precise predictions of mobile phone costs. This technology can alter how consumers make purchase decisions and allow businesses to improve their pricing strategies in the dynamic and cutthroat mobile phone industry by solving the aforementioned restrictions and taking a wider variety of factors into account.

Author Contributions: Conceptualization, P.-F.P.; data curation; W.-C.W.; formal analysis, P.-F.P. and W.-C.W.; funding acquisition, P.-F.P.; methodology, P.-F.P. and W.-C.W.; software, W.-C.W.; visualization, P.-F.P.; writing-original draft, review and editing, P.-F.P. All authors have read and agreed to the published version of the manuscript.

Note: The contributions mentioned above are indicative and may vary depending on the specific roles and expertise of the authors. All authors made intellectual contributions and collaborated closely to produce a comprehensive and cohesive research project on predicting mobile phone prices using machine learning techniques and transaction data analysis.

Title: Predicting Mobile Phone Prices: An Analysis of Transaction Data Using Machine Learning

Principal Investigator: [Gao]

Affiliation: [university of Essex]

Amount Requested:

Duration: 12 months

Conflicts of Interest: The authors declare no conflict of interest

References

Alqadi, Z. (2021). Artificial Neural Networks (ANN) [Online]. Available at: https://www.researchgate.net/publication/348150890_Artificial_neural_networks_ANN (Accessed: 3 May 2023)

Al-Mejibli, I.S., Alwan, J.K. and Abd Dhafar, H., 2020. The effect of gamma value on support vector machine performance with different kernels. *International Journal of Electrical and Computer Engineering*, *10*(5), p.5497.

Brieman, L. (2001). ‘Random Forests’, Machine Learning, 45(1), pp. 5 – 32. doi: 10.1023/A:1010950718922. (Accessed: 21 April 2023)

Chamasemani, F. F. and Singh, Y. P. (2011). ‘Multi-class Support Vector Machine (SVM) classifiers – An Application in Hypothyroid detection and Classification’, 6th International Conference on Bio-Inspired Computing: Theories and Applications [Online]. Available at: https://www.researchgate.net/publication/221608588_Multi-class_Support_Vector_Machine_SVM_Classifiers_--_An_Application_in_Hypothyroid_Detection_and_Classification/link/5fbe807e458515b7976ed8a8/download (Accessed: 1 May 2023)

Cover, T. M. and Hart, P. E. (1967). ‘Nearest Neighbor Pattern Classification’, IEEE Transactions on Information Theory, 13(1) pp. 21 – 27 [Online]. Available at: https://www.google.com/url?sa=t&source=web&rct=j&url=https://isl.stanford.edu/~cover/papers/transIT/0021cove.pdf&ved=2ahUKEwiJ_u_riff-AhUDYsAKHQIgBOYQFnoECBMQAQ&usg=AOvVaw1Bct0mFR5PViBKIAQLNEWk (Accessed: 25 April 2023)

Cutler, A., Stevens J. R. and Cutler, D. R. (2011). ‘Random Forests’, Machine Learning, [Online]. Available at: https://www.researchgate.net/publication/236952762_Random_Forests/link/0a85e52f509178323a000000/download (Accessed: 24 April 2023)

Ho, T. K. (1995). ‘Random decision forests’, Proceedings of 3rd International Conferenceon Document Analysis and Recognition, 1, pp. 278 – 282 [Online]. Available at: https://www.google.com/url?sa=t&source=web&rct=j&url=http://vision.cse.p

su.edu/seminars/talks/2009/random_tff/odt.pdf%3Forigin%3Dpublication_detail&ved=2ahUKEwiwrMXAhvf-AhVlR0EAHV6vBXoQFnoECEoQAQ&usg=AOvVaw0HSkYczp_fI19R6NT_yQPL (Accessed: 26 April 2023)

Imam, T., Ting, K.M. and Kamruzzaman, J., 2006. z-SVM: An SVM for improved classification of imbalanced data. In *AI 2006: Advances in Artificial Intelligence: 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006. Proceedings 19* (pp. 264-273). Springer Berlin Heidelberg.

Imandoust, S. B. and Bolandraftar, M. (2013). 'Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background', Journal of Engineering Research and Applications, 3(5), pp. 605 – 610 [Online]. Available at: https://www.researchgate.net/publication/304826093_Application_of_K-nearest_neighbor_KNN_approach_for_predicting_economic_events_theoretical_background (Accessed: 27 April 2023)

Pai, P and Wang W. (2020). 'Using Machine Learning Models and Actual Transaction Data for Predicting Real Estate Prices', Applied Sciences, 10(17). doi: 10.3390/app10175832. (Accessed: 20 April 2023)

Learn Mobile Price Prediction Through Classification Algorithms (analyticsvidhya.com)

Quinlan, J. R. (1986). ‘Induction of decision trees’, Machine learning, 1, pp. 81 – 106 [Online]. Available at: https://www.google.com/url?sa=t&source=web&rct=j&url=https://hunch.net/~coms-4771/quinlan.pdf&ved=2ahUKEwj9i9XU4fn-AhUJaMAKHZ_JAzkQFnoECBwQAQ&usg=AOvVaw3qgri751Lj_Wd9a8_uy2dd (Accessed: 21 April 2023)

Sekhar, C. and Meghana P. S. (2020) ‘A study on backpropagation in Artificial Neural Networks’, Asia-Pacific Journal of Neural Networks and its Applications, 4(1), pp.21-28 [Online]. Available at: https://www.researchgate.net/publication/349077282_A_Study_on_Backpropagation_in_Artificial_Neural_Networks/link/6137600e38818c2eaf88936f/download (Accessed: 5 May 2023)

Xuefeng, L., Lu, L., Lihua, W. and Zhao, Z., 2006. Predicting the final prices of online auction items. *Expert Systems with Applications*, *31*(3), pp.542-550.

Wikipedia, 2023. Support Vector Machine [Online] Available at: https://en.m.wikipedia.org/wiki/Support_vector_machine (Accessed: 28 April 2023)

Image: https://blakelobato1.medium.com/k-nearest-neighbor-classifier-implement-homemade-class-compare-with-sklearn-import-6896f49b89e