A REPORT ON
UNDERSTANDING OF GOOGLE PLAYSTORE APPS

This project dissertation is submitted to Osmania University, as

Partial fulfilment for the Completion of M.Sc. (Statistics) III-Semester

Sirisha Gani	1007-23-507-002
Akkenapalli Mandira	1007-23-507-014
Sanagonda Shylaja	1007-23-507-026
Sk Israk Sahan	1007-23-507-032
Ayesha Ruhi	1007-23-507-038

Under the supervision of

Prof. G. Jayasree

(**Head Department of Statistics**)

Department of Statistics
University College of Science
Osmania University, Hyderbad-500007

DECLARATION

We hereby declare that the project work presented in this dissertation, entitled **"A Report On Understanding Of Goggle Playstore Apps"** has been carried out by Sirisha Gani, Akkenapalli Mandira, Sanagonada Shylaja, Sk Israk Sahan, Ayesha Ruhi under the supervision of **Prof. G. Jayasree, Head**, in the Department of Statistics, University College of Science, Osmania University, Hyderabad: 500 007. The work done is original and has not been submitted so far, in part or full, for any other degree or diploma to any University or institution.

1. Shirisha Gani :1007-23-507-002
2. Akkenapalli Mandira :1007-23-507-014
3. Sanagonda Shylaja :1007-23-507-026
4. Sk Israk Sahan :1007-23-507-026
5. Ayesha Ruhi :1007-23-507-038

Date:

Department of Statistics
University College of Science
Osmania University
Hyderabad-500007.

Date:

CERTIFICATE

This is to certify that the research work presented in this thesis, entitled **"A Report On Understanding Of Google Playstore Apps"** has been carried out by Sirisha Gani, Akkenapalli Mandira, Sanagonada Shylaja, Sk Israk Sahan, Ayesha Ruhi registered M.Sc. (**Statistics**) III-Semester students, of this University. The work done is original and has not been submitted so far, in part or full, for any other degree or diploma to any University or institution.

Date: Signature of the Supervisor

Signature of Head with office Seal

Date of Examination:

Internal Examiner Signature with date:

External Examiner Signature with date:

Acknowledgment

We are thankful to our supervisor **Prof. G. Jayasree, Head, Department of Statistics, University College of Science, Osmania University, Hyderabad- 500 007,** for her guidance.

We are also thankful to **Prof. G. Jayasree, Head, Department of Statistics, University College of Science, Osmania University, Hyderabad- 500 007**. for providing necessary facilities for doing this project.

We also would like to express our sincere thanks to our professors**:**

Prof. N.Ch. Bhatracharyulu, CBOS, Prof. S.A. Jyothi Rani, Dr. G. Sirisha, Dr. M. Raghavender Sharma, and Mrs. J.L. Padmashree, Department of Statistics, Osmania University for their encouragement throughout the period of our course work.

We extend our sincere thanks to all non-teaching staff, Department of Statistics, for their secretarial help during the period.

1. Shirisha Gani :1007-23-507-002
2. Akkenapalli Mandira :1007-23-507-014
3. Sanagonda Shylaja :1007-23-507-026
4. Sk Israk Sahan :1007-23-507-032
5. Ayesha Ruhi :1007-23-507-038

CHAPTER 1

INTRODUCTION

CHAPTER-1

INTRODUCTION

1.1. Introduction

The Google Play Store is one of the largest and most popular Android app stores, offering an enormous amount of data that can be used to develop an optimal predictive model. We utilized a raw dataset from the Kaggle website, which contains 12 different features that can help determine whether an app will be successful.

Mobile applications represent one of the fastest-growing segments of the downloadable software market. Among the various platforms available, we chose the Google Play Store due to its increasing popularity and rapid growth. One key factor contributing to this popularity is that approximately 81% of the apps are free of cost. This rapid growth has led to over 500 million users downloading nearly 40 billion apps worldwide. Both developers and users play a crucial role in shaping the future of app markets through their interactions. However, the lack of a clear understanding of the inner workings and dynamics of these markets impacts both groups.

Importance

- The Google Play Store provides users with access to a vast collection of applications developed using the Android Software Development Kit (SDK).
- In addition to apps, it offers a variety of digital media for purchase, including books, movies, music, television programs, and video games.
- The Play Store comes pre-installed on many Android devices, ensuring seamless access for users. Furthermore, it is also available on select Chromebooks, expanding its usability beyond smartphones and tablets.

1.2. Data Description

The dataset is extracted from the source www.kaggle.com. and has been utilised for the predictions of Google Playstore of Apps using different methods of Machine Learning.

Usage: Google Playstore Apps

Format: A data frame with 10,842 observations following with the 12 variables.

We used the features like,

- Apps: Name of the applications.
- Category: Category the app belongs (e.g., games, productivity, etc.).
- Rating: Average user rating for the app (range: 0 to 5).
- Reviews: Number of user reviews for the app.
- Size: Size of the app (e.g., “19M” for “19MB”).
- Installs: Number of installs, formatted as ranges (e.g., “1,000,000+”).
- Type: Whether the app is Free or Paid.
- Price: Price of the app (in USD), applicable only for paid apps.
- Content Rating: Age group the app is suitable for (e.g., “Everyone”, “Teen”, “Adult”).
- Genres: Genre of the app, which could be more specific than the category.
- Current Ver: Current version of the app.
- Android Ver: Minimum Android OS version required to run the app (e.g., "4.0 and up").

1.3. Sample data

The following is the sample data set taken from the data frame of 10,842 observations using the “head()”.

The data sample below consists of the first five observations of the dataframe which is being studied.

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Ra	Genres	Last Updat	Current V	Android V
Photo Editor & C Camera & Grid & ScrapBook	ART_AND_DESIGN			19M	10,000+	Free		Everyone	Art & Design		1.0.0	4.0.3 and
Coloring book m	ART_AND_DESIGN			14M	500,000+	Free		Everyone	Art & Design;Prete		2.0.0	4.0.3 and
U Launcher Lite Live Cool Theme Apps	ART_AND_DESIGN			8.7M	5,000,000+	Free		Everyone	Art & Design		1.2.4	4.0.3 and
Sketch - Draw &	ART_AND_DESIGN			25M	50,000,000+	Free		Teen	Art & Design		Varies wi	4.2 and u
Pixel Draw - Nur Coloring Book	ART_AND_DESIGN			2.8M	100,000+	Free		Everyone	Art & Design;Creat			4.4 and u

Even though our dataset has 10,842 data-point, we make use of 7418 rows x 10 columns, due to removing duplicated values, missing values, and outliers.

1.4. Objectives of the project

Objective of Google Playstore Apps

Our main objective is to provide developers and Marketers with the insights of the Android app market and its Trends, for that we are using:

- Data visualization
- Descriptive Statistics
- Comparative Analysis
- Group Analysis
- Predictive Analysis using Machine Learning

CHAPTER 2
DATA MODELING THROUGH MACHINE LEARNING TECHNIQUES

CHAPTER-2

DATA MODELING THROGH MACHINE LEARNING TECHNIQUES

2. INTRODUCTION

2.1. What is Machine Learning?

Machine learning (ML) is a branch of artificial intelligence (AI) focused on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data.

Fig2

How Machine Learning Works?

Machine learning is a type of artificial intelligence (AI) that uses algorithms to learn from data and make predictions. Machine learning models can learn to identify patterns, recognize objects, and make decisions. Each step is important and cannot be skipped to achieve high accuracy. Code Implementation follows the below-mentioned steps:

- Data Collection
- Data Preprocessing
- Model Training
- Model Evaluation
- Model Deployment

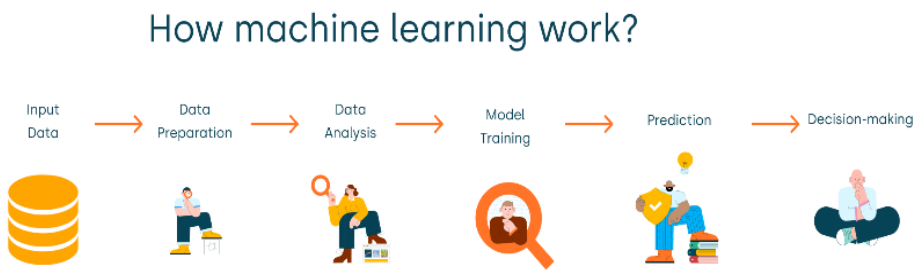


Fig3

1. **Supervised Learning**: Supervised learning is a type of machine learning where the algorithm learns from labelled data, meaning each training example consists of input features along with corresponding output labels. The algorithm learns to map input features to the output labels by finding patterns and relationships in the data. This allows it to make predictions or decisions on new, unseen data based on the learned patterns.

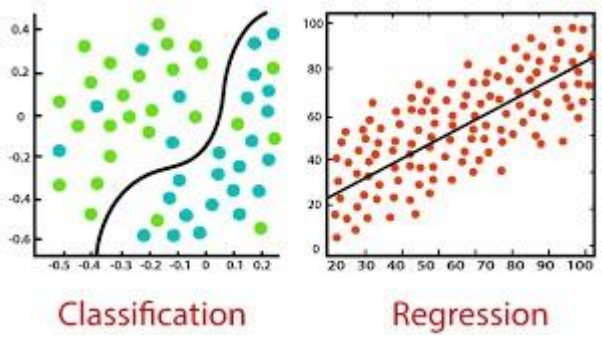


Fig 4

Supervised learning tasks include classification, where the algorithm predicts a discrete label or category, and regression, where it predicts a continuous value.

Regression Algorithm	Classification Algorithm
The output variable has to be a real value or continuous in nature.	The output variable is discrete in the Classification SML problem.
Regression algorithm helps to map the input value and the continuous output variable.	Classification algorithm helps in mapping the input value with the output variable which is discrete in nature.
Regression algorithms are only used for data that is continuous.	Classification algorithms are only used for data that is discrete.
Finds the best fit for accurately predicting the output data in Regression.	Finds the decision boundary which helps in classifying the dataset into different classes in Classification
Regression algorithms help in solving the problems like predicting the weather or the price of Houses in a neighbourhood according to a real estate trend.	Classification algorithms are useful for problems like identifying spam emails, recognition of speech, identifying cancer cells, etc.

2.Unsupervised Learning: Unsupervised learning is a type of machine learning where the algorithm learns from unlabelled data, meaning it doesn't have explicit output labels associated with the input features. Instead, the algorithm identifies patterns, structures, or relationships within the data without any guidance.

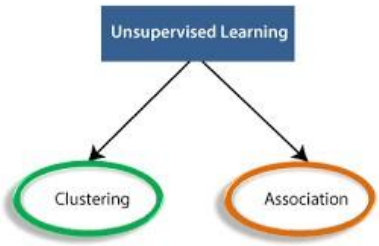


Fig 5

Unsupervised learning tasks include clustering, where the algorithm groups similar data points together, and dimensionality reduction, where the algorithm reduces the number of features while preserving important information.

3.Semi-supervised Learning: Semi-supervised learning is a type of machine learning that combines both labelled and unlabelled data for training. This approach is useful when labelled data is scarce or expensive to obtain. The algorithm learns from the small amount of labelled data available and utilizes the larger pool of unlabelled data to improve its performance.

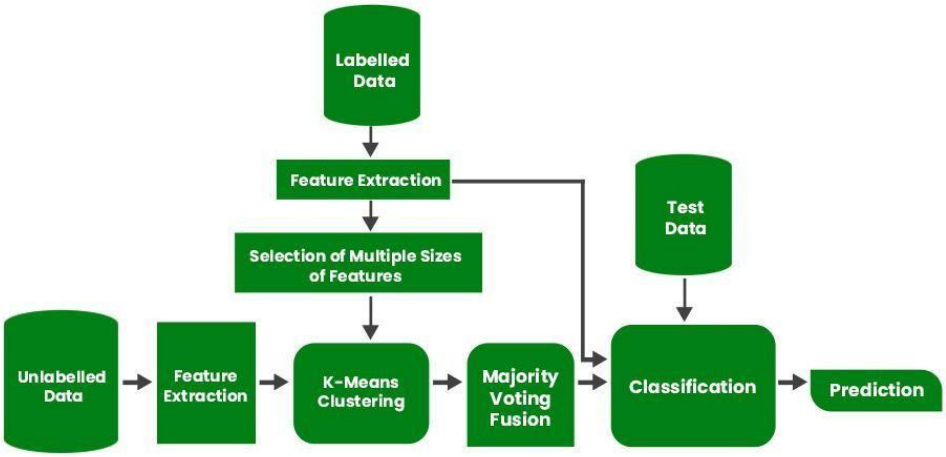


Fig 6

Semi-supervised learning algorithms typically leverage the structure or distribution of the unlabelled data to generalize better to new, unseen examples. This approach can lead to improved model performance compared to using only labelled data, especially in scenarios where obtaining labelled data is challenging.

4. Reinforcement learning:

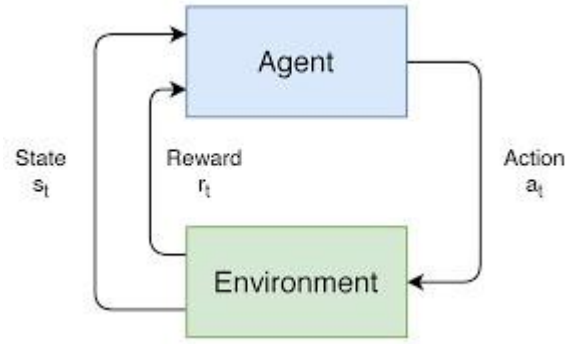


Fig 7

Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The goal is to maximize the total rewards accumulated over time.

2.1 Importance of Machine Learning

Machine learning is important because it gives enterprises a view of trends in customer behaviour and operational business patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google, and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

Machine learning has several practical applications that drive the kind of real business results - such as time and money savings - that have the potential to dramatically impact the future of your organization. In particular, we see tremendous impact occurring within the customer care industry, whereby machine learning is allowing people to get things done more quickly and efficiently. Through Virtual Assistant solutions, machine learning automates tasks that would otherwise need to be performed by a live agent - such as changing a password or checking an account balance. This frees up valuable agent time that can be used to focus on the kind of customer care that humans perform best: high touch, complicated decision-making that is not as easily handled by a machine. At Interactions, we further improve the process by eliminating the decision of whether a request should be sent to a human or a machine: unique Adaptive Understanding technology, the machine learns to be aware of its limitations, and bailout to humans when it has low confidence in providing the correct solution.

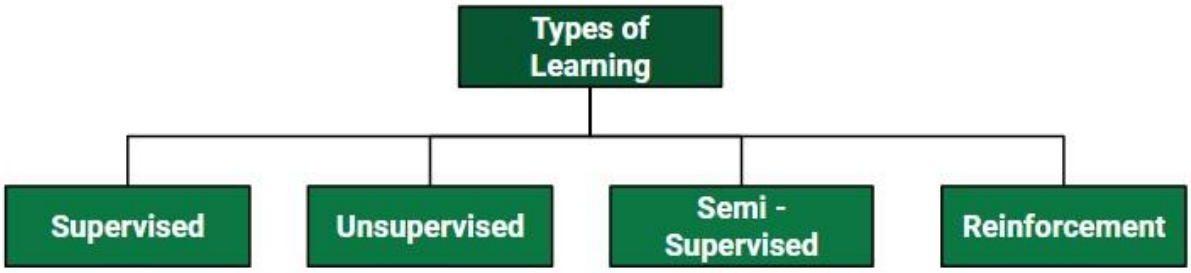


Fig 8

Some important applications in which machine learning is widely used are given below

1. Healthcare: Machine Learning is widely used in the healthcare industry. It helps healthcare researchers to analyze data points and suggest outcomes. Natural language processing helped to give accurate insights for better results of patients. Further, machine learning has improved the treatment methods by analyzing external data on patients' conditions in terms of X-ray, Ultrasound, CT-scan, etc. NLP, medical imaging, and genetic information are key areas of machine learning that improve the diagnosis, detection, and prediction system in the healthcare sector.

2. Automation: This is one of the significant applications of machine learning that helps to make the system automated. It helps machines to perform repetitive tasks without human intervention. As a machine learning engineer and data scientist, you have the responsibilities to solve any given task multiple times with no errors. However, this is not practically possible for humans. Hence machine learning has developed various models to automate the process, having the capability of performing iterative tasks in lesser time.

3. Banking and Finance: Machine Learning is a subset of AI that uses statistical models to make accurate predictions. In the banking and finance sector, machine learning helped in many ways, such as fraud detection, portfolio management, risk management, chatbots, document analysis, high-frequency trading, mortgage underwriting, AML detection, anomaly detection, risk credit score detection, KYC processing, etc. Hence, machine learning is widely applied in the banking and finance sector to reduce error as well as time.

4. Transportation and Traffic Prediction: This is one of the most common applications of Machine Learning that is widely used by all individuals in their daily routine. It helps to ensure highly secured routes, generate accurate ETAs, predict vehicle breakdown, Driving Prescriptive Analytics, etc. Although machine learning has solved transportation problems, it still requires more improvement. Statistical machine learning algorithms helps to build a smart transportation system. Further, deep Learning explored the complex interactions of roads, highways, traffic, environmental elements, crashes, etc. Hence, machine learning technology has improved daily traffic management as well as a collection of traffic data to predict insights of routes and traffic.

5. Image Recognition: It is one of the most common applications of machine learning which is used to detect the image over the internet. Further, various social media sites such as Facebook

uses image recognition for tagging the images to your Facebook friends with its feature named auto friend tagging suggestion.

Further, now a day's, almost all mobile devices come with exciting face detection features. Using this feature, you can secure your mobile data with face unlocking, so if anyone tries to access your mobile device, they cannot open without face recognition.

6. Speech Recognition: Speech recognition is one of the biggest achievements of machine learning applications. It enables users to search content without writing text or, in other words, 'search by voice'. It can search content/products on YouTube, Google, Amazon, etc. platforms by your voice. This technology is referred to as speech recognition.

It is a process of converting voice instructions into the text; hence it is also known as 'Speech to text' or 'Computer speech recognition. Some important examples of speech recognitions are Google assistant, Siri, Cortana, Alexa, etc.

7. Product Recommendation: It is one of the biggest achievements made by machine learning which helps various e-commerce and entertainment companies like Flipkart, Amazon, Netflix, etc., to digitally advertise their products over the internet. When anyone searches for any product, they start getting an advertisement for the same product while internet surfing on the same browser.

This is possible by machine learning algorithms that work on users' interests or past experience and accordingly recommend them for products. For e.g., when we search for a laptop on the Amazon platform, then it also gets started with so many other laptops having the same categories and criteria. Similarly, when we use Netflix, we find some recommendations for entertainment series, movies, etc. Hence, this is also possible by machine learning algorithms.

8. Virtual Personal Assistance: This feature helps us in many ways, such as searching content using voice instruction, calling a number using voice, searching contact in your mobile, playing music, opening an email, Scheduling an appointment, etc. Now a day, you all have seen advertising like "Alexa! Play the Music" this is also done with the help of machine learning. Google Assistant, Alexa, Cortana, Siri, etc., are a few common applications of machine learning. These virtual personal assistants record our voice instructions, send them over to the server on a cloud, decode it using ML algorithms and act accordingly.

9. Email Spam and Malware detection & Filtering: Machine learning also helps us for filtering emails in different categories such as spam, important, general, etc. In this way, users can easily identify whether the email is useful or spam. This is also possible by machine learning algorithms such as Multi-Layer Perceptron, Decision tree, and Naïve Bayes classifier. Content filter, header filter, rules-based filter, permission filter, general blacklist filter, etc., are some important spam filters used by Google.

10. Self-driving cars: This is one of the most exciting applications of machine learning. Machine learning plays a vital role in the manufacturing of self-driving cars. It uses an unsupervised learning method to train car models to detect people and objects while driving. Tata and Tesla are the most popular car manufacturing companies working on self-driving cars. Hence, it is a big revolution in a technological era which is also done with the help of machine learning.

11. Credit card fraud detection: Credit card frauds have become very easy targets for online hackers. As the culture of online/digital payments is increasing, the risk of credit/debit cards is parallel increasing. Machine Learning also helps developers to detect and analyze frauds in online transactions. It develops a novel fraud detection method for Streaming Transaction Data, with an objective to analyze the past transaction details of the customers and extract the behavioral patterns. Further, cardholders are clustered into various categories with their transaction amount so that the behavioral pattern of the groups can be extracted respectively. Hence, credit card fraud detection is a novel approach using Aggregation Strategy and Feedback Mechanism of machine learning.

12. Stock Marketing and Trading: Machine learning also helps in the stock marketing and trading sector, where it uses historical trends or past experience for predicting the market risk. As share marketing is another name of marketing risk, machine learning reduces it to some extent and predicts data against marketing risk. Machine learning's long short-term neural memory network is used for the prediction of stock market trends.

13. Language Translation: The use of Machine learning can be seen in language translation. It uses the sequence-to-sequence learning algorithms for translating one language into other. Further, it also uses images recognition techniques to identify the text from one language to other. Similarly, Google's GNMT (Google Neural Machine Translation) provides this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it is called automatic translation.

2.2 Data Analytics using Python:

Python has become a staple in data science, allowing data analysts and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyse data. and complete other datarelated tasks.

Python can build a wide range of different data visualizations, like line and bar graphs, pie charts, histograms, and 3D plots: Python also has a number of libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently, like TensorFlow and Keras.

There are various library modules used in the python programming language for the purpose of data analysis which includes modules like Numpy, Pandas, Scikit learn, Matplotlib, seaborn, tensorflow, missingno etc. These libraries ease the user with the data handling, processing and as well as testing the same

Key topics:

- Installing Python/Jupyter/Python on Windows Or Mac
- Python Basics (variables, strings, simple math, conditional logic, for loops, lists, tuples, dictionaries, etc.)
- Using the Pandas library to manipulate data (filtering and sorting data, combining files, GroupBy, etc.)
- Plotting data in Python using Matplotlib and Seaborn
- Multiple Linear Regression using Scikit-Learn
- Decision Trees using Scikit-Learn
- Random Forests (Scikit-Learn)

2.3 Machine learning Techniques

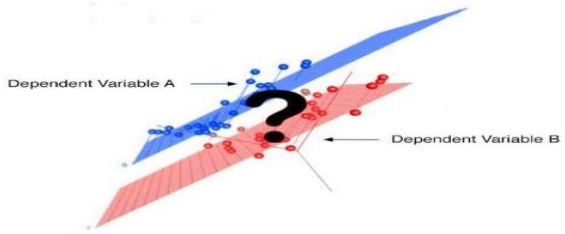
For fulfilling the objective of our Project as have used various techniques other than the government method for the prediction of the Google Playstore Apps. We have tested the data frame with different methods and tested the accuracy using different Statistical tools and obtained the efficient Machine Learning technique for the prediction of Google Playstore Apps with the variables we have tested. The Implemented methods are explained as follows

2.3.1 Multiple Linear Regression

Simple Linear Regression, where a single Independent/Predictor(X) variable is used to model the response variable (Y). But there may be various cases in which the response variable is affected by more than one predictor variable; for such cases, the Multiple Linear Regression algorithm is used.

Moreover, Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. We can define it as ***Multiple Linear Regression*** is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form. Each feature variable must model the linear relationship with the dependent variable. MLR tries to fit a regression line through a multidimensional space of data-points.



MLR equation:

In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables X1, X2, X3, Xn. Since it is an enhancement of Simple Linear Regression, so the same is applied for the multiple linear regression equation, the equation becomes:

$$Y=\beta0+\beta1X1+\beta2X2+\beta3X3+\cdots+\beta nXn+\epsilon$$

- Y = Dependent variable (target variable)
- X1 ,X2,...,Xn = Independent variables (predictors/features)
- β0 = Intercept (constant term)
- β1, β2,...,βn = Coefficients for each independent variable
- ϵ = Error term

Steps Involved in any Multiple Linear Regression Model

Step #1: Data Pre Processing

1. Importing The Libraries.
2. Importing the Data Set.
3. Encoding the Categorical Data.

4. Avoiding the Dummy Variable Trap.
5. Splitting the Data set into Training Set and Test Set.

Step#2: Fitting Multiple Linear Regression to the Training set
Step#3: Predict the Test set results.

Assumptions for Multiple Linear Regression:

- **Linearity:** The relationship between dependent and independent variables should be linear.
- **Homoscedasticity:** Constant variance of the errors should be maintained.
- **Multivariate normality:** Multiple Regression assumes that the residuals are normally distributed.
- **Lack of Multicollinearity:** It is assumed that there is little or no multicollinearity in the data.

It performs a regression task. Regression models are target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Some of the advantages and disadvantages are stated below. Here are the few stated Pros and Cons of the Multi-Linear Regression.

Advantages	Disadvantages
Linear Regression is simple to implement and to interpret the output coefficients.	On the other hand in linear regression tec outliers can have huge effects on the regressi boundaries are linear in this technique.
When you know the relationship betwe independent and dependent variable have a relationship, this algorithm is the best to use b of it's less complexity to compared to algorithms.	Diversely, linear regression assumes a relationship between dependent and indep variables. That means it assumes that ther straight-line relationship between them. It a independence between attributes.
Linear Regression is susceptible to over- fittin can be avoided using some dimensionality rec techniques, regularization (LI and L2) techniq cross-validation.	Diversely, linear regression assumes a relationship between dependent and indep variables. That means it assumes that ther straight-line relationship between them. It a independence between attributes.

2.3.2 Decision Tree Regressor:

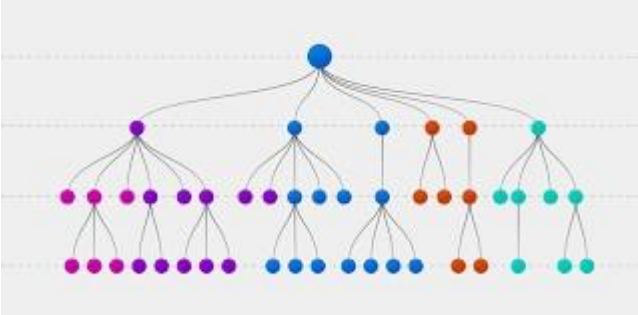


Fig 11

A Decision Tree Regressor is a machine learning algorithm used for regression tasks. It works by partitioning the feature space into a set of rectangular regions and fitting a simple model (usually a constant value) to each region. This allows it to predict continuous values for new data points based on their feature values.

Assumptions for Decision Tree Regressor:

Decision tree regressors do not rely on strict assumptions like some other statistical models do (e.g., linear regression). However, they can suffer from overfitting if not properly regularized, and they may not perform well with datasets that have high dimensionality or noisy features. Additionally, decision trees assume that the relationships between features and target variables are captured by the tree's splits, which may not always be the case in complex datasets. Regularization techniques like pruning or using ensemble methods like Random Forests can help mitigate these issues.

Advantages:

1. **Interpretability:** Decision trees are easy to understand and interpret, making them suitable for explaining the model's decisions to non-technical stakeholders.

2. **Non-parametric:** They do not make any assumptions about the distribution of data, which allows them to capture complex relationships between features and target variables.

3 **Handles non-linear relationships**: Decision trees can capture non-linear relationships between features and target variables effectively.

4. **Handles both numerical and categorical data:** Decision trees can handle both numerical and categorical features without the need for one-hot encoding or other transformations.

5. **Robust to outliers**: Decision trees are robust to outliers and missing values, as they make decisions based on splitting data into different regions.

Disadvantages:

1. **Overfitting**: Decision trees are prone to overfitting, especially when the tree depth is not properly controlled or when the dataset is noisy.

2. **Instability:** Small variations in the data can lead to significantly different trees, making decision trees unstable and sensitive to changes in the training data.

3. **High variance:** Decision trees can have high variance, meaning they may perform poorly on new, unseen data if they have not been properly regularized or if the training data is not representative of the underlying distribution.

4. **Bias towards features with more levels:** Decision trees tend to favor features with more levels (i.e., more possible splits), which can lead to biased predictions if some features dominate the splits.

Not suitable for extrapolation: Decision trees are not suitable for extrapolation, meaning they may not perform well when making predictions outside the range of values seen in the training data.

2.3.3 Random Forest Regressor:

Random Forest Regressor is a popular machine learning algorithm used for regression tasks. It works by constructing multiple decision trees during training and outputs the mean prediction of the individual trees for regression problems. It's effective for handling nonlinear relationships and is robust to overfitting.

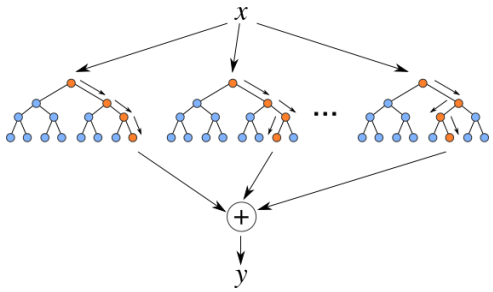


Fig 10

Assumptions for Random Forest Regressor:

1. **Independence of features**: Random Forest assumes that the features used for training the model are independent of each other. Violation of this assumption might lead to biased model predictions.

2. **No multicollinearity**: It assumes that there is no multicollinearity among the predictor variables. Multicollinearity occurs when two or more predictor variables in a regression model are

highly correlated. This can lead to unstable estimates of the coefficients and affect the model's performance.

3. **Homoscedasticity**: Random Forest does not explicitly assume homoscedasticity, which means that the variance of the errors is constant across all levels of the independent variables. However, it is robust to violations of this assumption due to its ensemble nature.

4. **Large dataset:** While Random Forest can perform well on smaller datasets, it tends to perform better with larger datasets due to its ability to handle high-dimensional data and capture complex relationships.

Pros:

1. **High Accuracy**: Random Forest tends to provide high accuracy in both classification and regression tasks. It can capture complex nonlinear relationships in the data, making it suitable for a wide range of problems.

2. **Robustness to Overfitting**: By aggregating the predictions of multiple decision trees, Random Forest reduces the risk of overfitting compared to individual decision trees. It achieves this by averaging or voting on the predictions of the ensemble.

3. **Feature Importance**: Random Forest provides a measure of feature importance, which helps in understanding the relative contribution of each feature to the prediction. This can be valuable for feature selection and understanding the underlying data patterns.

4. **Handles Missing Values and Outliers**: Random Forest can handle missing values and outliers in the data without preprocessing, making it convenient to use, especially in real world datasets where data quality may vary.

Cons:

1. **Black Box Nature**: Random Forest is often considered a "black box" model, meaning it lacks interpretability compared to simpler models like linear regression. Understanding the internal workings of the model and explaining predictions can be challenging.

2. **Memory and Computational Resources:** Random Forests can be memory-intensive, especially when dealing with large datasets or a large number of trees in the ensemble. Additionally, predicting with a large number of trees can be computationally expensive.

3. **Hyperparameter Tuning:** Random Forest has several hyper parameters that need to be tuned to achieve optimal performance, such as the number of trees, tree depth, and minimum samples per leaf. Finding the right set of hyperparameters can require extensive experimentation.

4. **Bias Towards Majority Class**: In classification tasks with imbalanced class distributions, Random Forest may exhibit a bias towards the majority class, leading to suboptimal performance for minority classes unless addressed through techniques like class weighting or resampling.

5. **Parallelization**: Training multiple decision trees in a Random Forest can be easily parallelized, leading to faster training times compared to other algorithms, especially on multicore processors or distributed computing environments.

2.4 Model Evaluation Metrics

Model evaluation metrics are essential for assessing the performance of a machine learning model. Depending on the type of problem (classification, regression, etc.), different metrics are used. Here's a summary of the most common evaluation metrics:

1. Classification Models Evaluation Metrics:

For classification tasks, models predict a discrete class label, and evaluation focuses on how well the model can differentiate between these classes. Key metrics include:

Accuracy

- **Definition**: The proportion of correct predictions (both positive and negative) over all predictions.
- **Formula**: (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False Negatives)

Interpretation: Accuracy is intuitive but can be misleading when dealing with imbalanced datasets (where one class significantly outnumbers the other). For example, in a dataset with 95% of class A and 5% of class B, a model predicting only class A would have 95% accuracy but would fail to identify class B.

Precision: The proportion of positive predictions that are actually correct.

Precision= True Positives / (True Positives + False Positives)

Interpretation: Precision is important when the cost of a false positive is high (e.g., email spam detection, where a false positive means a valid email is marked as spam). High precision means fewer false positives.

Recall: **T**he proportion of actual positive instances that are correctly identified.

Recall= True Positives / (True Positives + False Negatives)

Interpretation: Recall is important when the cost of a false negative is high (e.g., in medical diagnostics where missing a positive case could have severe consequences). High recall means fewer false negatives.

F1-Score: The harmonic means of precision and recall, which balances both metrics.

F1-Score= 2 * (Precision * Recall) / (Precision + Recall)

Interpretation: F1-Score is useful when you need a balance between precision and recall and when the class distribution is imbalanced. It's particularly useful for binary classification problems.

2. Regression Models Evaluation Metrics:

In regression tasks, models predict continuous values. The following metrics help assess how well the predicted values match the actual values.

Mean Absolute Error (MAE)

Definition: The average of the absolute errors between predicted and actual values.

$$MAE = (1/n) * \Sigma |yi - \hat{y}i|$$

Where:

- **n** is the number of data points.
- **yi** is the actual value for the i-th data point.
- **ŷi** is the predicted value for the i-th data point.
- **|yi - ŷi|** is the absolute error for each data point.

Interpretation: MAE gives an easily interpretable value in the same units as the target variable. It's robust to outliers but doesn't penalize large errors as much as squared error-based metrics.

Mean Squared Error (MSE): The average of the squared differences between predicted and actual values.

$$MSE = (1/n) * \Sigma (yi - \hat{y}i)^2$$

Where:

- **n** is the number of data points.
- **yi** is the actual value for the i-th data point.
- **ŷi** is the predicted value for the i-th data point.
- **$(yi - \hat{y}i)^2$** is the squared error for each data point.

Interpretation: MSE penalizes larger errors more than smaller ones because the errors are squared. It's sensitive to outliers, and larger errors significantly influence the metric.

Root Mean Squared Error (RMSE): The square root of the mean squared error. It brings the error back to the same unit scale as the target variable.

$$RMSE = \sqrt{[(1/n) * \Sigma (yi - \hat{y}i)^2]}$$

Where:

- **n** is the number of data points.
- **yi** is the actual value for the i-th data point.
- **ŷi** is the predicted value for the i-th data point.
- **$(yi - \hat{y}i)^2$** is the squared error for each data point.

Interpretation: RMSE is more interpretable than MSE since it is in the same units as the target variable. Like MSE, it is sensitive to outliers, which makes it useful for situations where large errors are undesirable.

R-Squared (R^2): The proportion of the variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - [\Sigma (yi - \hat{y}i)^2 / \Sigma (yi - \bar{y})^2]$$

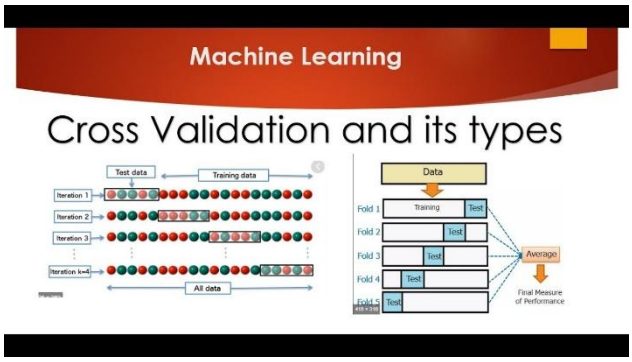
Where:

- **yi** is the actual value for the i-th data point.
- **ŷi** is the predicted value for the i-th data point.
- **ȳ** is the mean of the actual values.
- **$(yi - \hat{y}i)^2$** is the squared error of the prediction.
- **$(yi - \bar{y})^2$** is the squared variance from the mean of the actual values.

Interpretation: R^2 ranges from 0 to 1, with 1 indicating perfect predictions and 0 suggesting no explanatory power. A higher R^2 means the model is better at explaining the variance of the target variable

2.5 Cross Validation

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set.



Types of Cross-Validation:

1.Holdout Validation:

In Holdout Validation, we perform training on the 50% of the given dataset and rest 50% is used for the testing purpose. It's a simple and quick way to evaluate a model. The major drawback of this method is that we perform training on the 50% of the dataset, it may possible that the remaining 50% of the data contains some important information which we are leaving while training our model i.e. higher bias.

2. LOOCV (Leave One Out Cross Validation)

In this method, we perform training on the whole dataset but leaves only one data-point of the available dataset and then iterates for each data-point. In LOOCV, the model is trained on n−1n−1 samples and tested on the one omitted sample, repeating this process for each data point in the dataset. It has some advantages as well as disadvantages also.

An advantage of using this method is that we make use of all data points and hence it is low bias.

The major drawback of this method is that it leads to higher variation in the testing model as we are testing against one data point

.

3. Stratified Cross-Validation

It is a technique used in machine learning to ensure that each fold of the cross-validation process maintains the same class distribution as the entire dataset. This is particularly important when dealing with imbalanced datasets, where certain classes may be underrepresented. In this method,

1. The dataset is divided into k folds while maintaining the proportion of classes in each fold.
2. During each iteration, one-fold is used for testing, and the remaining folds are used for training.
3. The process is repeated k times, with each fold serving as the test set exactly once.

Stratified Cross-Validation is essential when dealing with classification problems where maintaining the balance of class distribution is crucial for the model to generalize well to unseen data.

4. K-Fold Cross Validation

In K-Fold Cross Validation, we split the dataset into k number of subsets (known as folds) then we perform training on the all the subsets but leave one (k-1) subset for the evaluation of the trained model. In this method, we iterate k times with a different subset reserved for testing purpose each time.

CHAPTER 3
SOFTWARE USED FOR DATA ANALYSIS

CHAPTER 3

SOFTWARE USED FOR DATA ANALYSIS

3.1 INTRODUCTION

3.1.1 Python Programming

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid at the Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program at the end maintenance. Python supports modules and packages, which encourages in program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed**.**

3.1.2 Importance of Python

Python's importance stems from its versatility, simplicity, and extensive ecosystem. As one of the most popular programming languages, Python facilitates rapid development across various domains, from web development and data science to artificial intelligence and automation. Its clear syntax and readability make it accessible to beginners while offering advanced features for experienced developers. With a vast array of libraries and frameworks, Python enables efficient problem-solving and fosters innovation. Its open-source nature fosters a collaborative community, driving continual improvement and adaptation. Overall, Python's significance lies in its power to streamline development processes, enhance productivity, and fuel technological advancements across industries.

3.1.3 Advantages of Python

Readability and simplicity: Python emphasizes readability and simplicity, which makes it easy to learn and understand, even for beginners. Its clean and straightforward syntax reduces the cost of program maintenance and development.

Large standard library: Python comes with a vast standard library that provides support for various tasks and functionalities, such as string operations, file I/O, networking, database interfaces, and much more. This extensive library helps developers accomplish complex tasks without having to write additional code from scratch.

Community and support: Python has a large and active community of developers, enthusiasts, and contributors who constantly work to improve the language. This vibrant community provides extensive support through forums, mailing lists, and online resources, making it easier for developers to find solutions to their problems and learn from others.

Portability: Python is a platform-independent language, meaning that Python code can run on various platforms and operating systems without any modifications. This portability makes it an ideal choice for developing cross-platform applications.

Flexibility and versatility: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Its versatility allows developers to choose the most appropriate approach for their projects and switch between paradigms as needed.

Integration capabilities: Python seamlessly integrates with other programming languages and technologies, such as C/C++, Java, .NET, and more. This interoperability allows developers to leverage existing codebases and libraries written in other languages, enhancing productivity and efficiency.

3.1.4 Disadvantages of Python

Performance: Python is an interpreted language, which means it tends to be slower than compiled languages like C or C++. This can be a disadvantage in situations where high performance is critical, such as in real-time systems or high-frequency trading applications. While there are ways to optimize Python code (e.g., using libraries like NumPy or Cython), it may still not match the performance of compiled languages.

GIL (Global Interpreter Lock): Python has a Global Interpreter Lock, which means that only one thread can execute Python bytecode at a time. This can limit the effectiveness of multi-threading for CPU-bound tasks since threads cannot fully utilize multiple CPU cores simultaneously.

Mobile Development: While Python can be used for mobile development using frameworks like Kivy or BeeWare, it's not as commonly used as languages like Java or Swift for building native mobile applications. This can be a disadvantage if you're primarily focused on mobile development.

Memory Consumption: Python's dynamic typing and high-level abstractions can lead to higher memory consumption compared to languages like C or C++. This can be a concern in memory-constrained environments, such as embedded systems or when dealing with large scale data processing tasks.

Packaging and Distribution: Python's packaging and distribution ecosystem can be complex and fragmented. While tools like pip and virtual environments help manage dependencies, dealing with different versions of packages, compatibility issues, and ensuring consistent environments across different systems can be challenging, especially for beginners.

3.2 DATA ANALYSIS USING PYTHON

3.2.1 Basic Statistics

Descriptive statistics refers to the branch of statistics that deals with summarizing and describing features of datasets. These statistics help in organizing, analyzing, and presenting data in a meaningful way, allowing researchers and analysts to understand and interpret the information effectively. Some common descriptive statistics include

1. **Measures of Central Tendency**:
    - **Mean:** The average value of the dataset, calculated by summing all values and dividing by the number of observations.
    - **Median:** The middle value of the dataset when it is arranged in ascending order.

- **Mode:** The value that occurs most frequently in the dataset.

2. **Measures of Dispersion**:
    - **Range:** The difference between the maximum and minimum values in the dataset.
    - **Variance:** A measure of how spread out the values in the dataset are from the mean.
    - **Standard Deviation:** The square root of the variance, indicating the average deviation of data points from the mean.
    - **Skewness:** A measure of the asymmetry of the distribution around its mean.
    - **Kurtosis:** A measure of the "tailedness" of the distribution, indicating whether the data are heavy-tailed or light-tailed relative to a normal distribution.
    - **Percentiles** divide the dataset into hundred equal parts, while quartiles divide it into four equal parts. They help in understanding the distribution of data and identifying outliers.

Correlation: Correlation measures the statistical relationship between two variables, indicating the extent to which changes in one correspond to changes in the other. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 suggesting no linear relationship. Correlation does not imply causation.

"Describe ()" method

The 'describe ()' method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains this information for each column:

Count- The number of non-empty values

Mean- The average(mean) value

Std- The standard deviation

Min- The minimum value

25%- The 25% percentile*

50%- The 50% percentile*

75%- The 75% percentile*

Max- The maximum value **Syntax:** Dataset name.describe ()

3.2.2 Data Visualisations

Pie Chart

A pie chart is a circular statistical graphic that represents data in a proportional manner, where each slice of the pie corresponds to a specific category or group of data. The size of each slice is proportional to the quantity it represents in relation to the whole dataset, typically expressed as percentages. Pie charts are useful for visually representing the distribution of a dataset and for comparing the relative sizes of different categories. They are commonly used to illustrate simple

proportions and are effective in conveying a quick overview of the composition of a dataset or the distribution of a categorical variable

Bar Chart

A bar chart is a graphical representation of data that uses rectangular bars of varying lengths to represent the frequency, count, or proportion of different categories or values in a dataset. Each bar typically corresponds to a specific category, with the length of the bar proportional to the magnitude of the data it represents. Bar charts are commonly used to compare and visualize discrete categories or groups of data, making it easy to identify patterns, trends, and relationships within the dataset. They are effective for presenting categorical data and are widely used in presentations, reports, and data visualization applications **Line Plot**

A line plot, also known as a line graph, is a type of data visualization that displays data points connected by straight lines. It typically consists of a horizontal x-axis representing independent variables (e.g., time, categories) and a vertical y-axis representing dependent variables (e.g., values, frequencies). Line plots are used to show trends or relationships between variables over a continuous interval. They are particularly useful for illustrating changes in data over time or comparing multiple datasets. Line plots are effective for conveying patterns and trends in a clear and concise manner, making them commonly used in various fields such as science, economics, and finance.

Histogram

A histogram is a graphical representation of the distribution of numerical data. It consists of a series of bars, where each bar represents the frequency or relative frequency of data within a certain range or interval. The horizontal axis typically represents the range of values, divided into bins or intervals, while the vertical axis represents the frequency or relative frequency of occurrences within each bin. Histograms provide a visual depiction of the shape, centre, and spread of a dataset, making it easy to identify patterns, outliers, and underlying distributions in the data. They are commonly used in exploratory data analysis and statistical inference.

Heat Map

A heatmap is a graphical representation of data where values in a matrix are represented as colors. Typically, a color gradient is used to indicate the magnitude of each value, with darker colors representing higher values and lighter colors representing lower values. Heatmaps are commonly used to visualize relationships or patterns in large datasets, such as correlations between variables or spatial distributions of data points. They provide an intuitive way to identify clusters, trends, and areas of interest within the data, making them valuable tools for data exploration, analysis, and presentation in various fields like biology, finance, and geography.

Pair Plot

A pair plot (scatterplot matrix) is a visualization tool used to explore relationships between multiple numerical variables in a dataset by plotting pairwise scatterplots for each variable combination. It helps in identifying trends, correlations, clusters, and potential outliers. The diagonal of the plot typically contains histograms or kernel density estimates (KDE) to show the distribution of individual variables. Pair plots are particularly useful for understanding data distributions, detecting

multicollinearity before regression modeling, and spotting anomalies. In Python, they can be created using Seaborn's `pairplot()` function, which allows for coloring points based on categorical variables using the `hue` parameter.

3.2.3 Packages used

Numpy

NumPy is a powerful Python library for numerical computing. It provides highperformance multidimensional array objects and tools for working with these arrays efficiently. Numpy's arrays facilitate mathematical operations on large datasets, making it popular in fields like data science, machine learning, and scientific computing. Additionally, NumPy offers a wide range of mathematical functions, linear algebra operations, and random number generation capabilities. Its efficient array operations and broadcasting capabilities make it a fundamental building block for many Python-based numerical computing tasks.

Pandas

Pandas is a powerful open-source Python library for data manipulation and analysis. It provides easy-to-use data structures like DataFrame and Series, which allow for efficient handling of structured data. With Pandas, users can perform operations such as filtering, grouping, sorting, and aggregation on datasets, making it a popular choice for data wrangling tasks. It also integrates well with other libraries in the Python ecosystem, such as NumPy and Matplotlib, enabling seamless data analysis and visualization workflows.

Matplotlib

Matplotlib is a popular Python library used for creating static, interactive, and animated visualizations. It offers a wide range of plotting functionalities for generating plots, charts, histograms, and more. Matplotlib provides a high degree of customization and flexibility, allowing users to create publication-quality graphics with ease. It integrates seamlessly with other libraries such as NumPy and Pandas, making it a powerful tool for data visualization and analysis in fields like data science, machine learning, finance, and scientific research.

Scikit-learn

Scikit-learn is a popular machine learning library in Python, providing a simple and efficient toolset for data mining and analysis. It offers various supervised and unsupervised learning algorithms, including classification, regression, clustering, dimensionality reduction, and model selection. Scikit-learn is built on NumPy, SciPy, and matplotlib, making it easy to integrate into existing Python data analysis workflows. Its user-friendly interface and extensive documentation make it a preferred choice for both beginners and experienced practitioners in the field of machine learning and data science.

Seaborn

Seaborn is a powerful Python visualization library built on top of Matplotlib, designed for creating aesthetically appealing and informative statistical graphics. It provides high-level functions for easily visualizing complex datasets, such as scatter plots, box plots, violin plots, and pair plots, making it particularly useful for exploratory data analysis. Seaborn integrates well with Pandas

DataFrames and includes built-in themes and color palettes to enhance readability. It also supports advanced statistical visualizations like regression plots and heatmaps for correlation analysis. By simplifying the process of data visualization, Seaborn helps analysts and data scientists gain deeper insights into their data with minimal code.

3.2.4 Machine Learning Methods

Linear regression

Syntax: from sklearn.linear_model import LinearRegression

```
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Decision tree

Syntax: from sklearn.tree import DecisionTreeClassifier

```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
accuracy = clf.score(X_test, y_test)
```

Random Forest

Syntax: from sklearn.ensemble import RandomForestClassifier or # Rgressor

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestClassifier()  # or RandomForestRegressor()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

CHAPTER 4 IMPLEMENTATION USING PYTHON

CHAPTER 4

IMPLEMENTATION USING PYTHON

4.1 Importing packages & Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt      importseaborn as sns
from sklearn.model_selection import train_test_split from sklearn.tree import Decision
TreeClassifier
from sklearn.ensemble import Random Forest Classifier from sklearn.metrics import
accuracy_score
import sklearn.metrics
data=pd.read_csv("googleplaystore.csv")
# importing dataset
```

4.2 Basic Statistical Analysis Code

Identifying duplicate values

```
df[df.duplicated()]
```

Identifying missing values

```
df.isnull().sum
```

Removing missing values

```
df.dropna()
```

Summarisation

```
df.info()
```

4.3 Data visualisation

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Histogram:

1. **# Plotting Rating distribution**

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Rating'].dropna(), bins=20, kde=True, color='blue')
plt.title('Distribution of App Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```

2. **# Plotting Apps size distribution**

```
plt.figure(figsize=(10,6))
sns.histplot(df['Size'].dropna(), kde=True, bins=20)
plt.title('App Size Distribution')
plt.xlabel('Size (KB)')
plt.ylabel('Frequency')
plt.show()
```

Scatter Plot

1. **# Price Vs Reviews**

```
plt.figure(figsize=(10,6))
sns.scatterplot(x='Reviews', y='Price', data=df)
plt.title('Price vs Reviews')
plt.ylabel('Price')
plt.xlabel('Number of Reviews')
plt.show()
```

2. **# Installs Vs Size**

```
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Size', y='Installs', data=df, hue='Rating', palette='viridis')
plt.title('Install vs Size')
plt.xlabel('Size (kB)')
plt.ylabel('Install')
plt.show()
```

Bar Plot

1. **# No. of Apps Vs Apps Category**

```
plt.figure(figsize=(12, 8))
```

```
category_counts = df['Category'].value_counts()
sns.barplot(x=category_counts.index, y=category_counts.values, palette='coolwarm')
plt.xticks(rotation=90)
plt.title('Number of Apps in Each Category')
plt.ylabel('Number of Apps')
plt.xlabel('Category')
plt.show()
```

2. **# Installs Vs Categories**

```
category_vs_installs = df.groupby('Category')['Installs'].sum().sort_values()
plt.figure(figsize=(12,8))
category_vs_installs.plot(kind='barh', color='skyblue')
plt.title('Total Installs by Category')
plt.xlabel('Total Installs')
plt.ylabel('Category')
plt.show()
```

Pie chart

```
app_type_counts = df['Type'].value_counts()
plt.figure(figsize=(7, 5))
plt.pie(app_type_counts, labels=app_type_counts.index, autopct='%1.1f%%',
colors=sns.color_palette('Set2', n_colors=2))
plt.title('Distribution of Free vs Paid Apps')
plt.show()
df_numeric = df.select_dtypes(include=['number'])
```

Correlation matrix

```
import matplotlib.pyplot as plt
import seaborn as sns
# Remove non-numeric columns
df_numeric = df.select_dtypes(include=['number'])
# Calculate the correlation matrix for the remaining numeric columns
plt.figure(figsize=(10, 6))
sns.heatmap(df_numeric.corr(), annot=True, cmap='coolwarm', fmt='.2f')
```

```
plt.title('Feature Correlation Heatmap')
plt.show()
```

Pair Plot

```
df_numeric = df.select_dtypes(include=['number'])
sns.pairplot(df_numeric)
plt.show()
```

4.4 Descriptive Statistics

```
  # Apply cleaning functions
df['Installs'] = df['Installs']
df['Size'] = df['Size']
df['Reviews'] = df['Reviews']
df['Price'] = df['Price']
# Descriptive Analysis
descriptive_summary = {
    "Rating Distribution": df['Rating'].describe(),
    "Category Popularity": df['Category'].value_counts(),
    "Price Analysis": df['Price'].describe(),
    "App Size Distribution": df['Size'].describe(),
    "Content Rating Distribution": df['Content Rating'].value_counts(),
    "Review Trends": df['Reviews'].describe(),
    "Genres Diversity": df['Genres'].value_counts(),
    "Android Version Requirements": df['Android Ver'].value_counts(),
}
print(descriptive_summary)
# Descriptive Analysis
rating_summary = df['Rating'].describe()
category_popularity = df['Category'].value_counts()
price_summary = df['Price'].describe()
size_summary = df['Size'].describe()
content_rating_summary = df['Content Rating'].value_counts()
review_summary = df['Reviews'].describe()
genres_diversity = df['Genres'].value_counts()
android_version_summary = df['Android Ver'].value_counts()
# Print
print("Rating Summary:\n", rating_summary)
print("\nCategory Popularity:\n", category_popularity)
print("\nPrice Summary:\n", price_summary)
print("\nSize Summary:\n", size_summary)
```

```
print("\nContent Rating Summary:\n", content_rating_summary)
print("\nReview Summary:\n", review_summary)
print("\nGenres Diversity:\n", genres_diversity)
print("\nAndroid Version Requirements:\n", android_version_summary)
```

4.5 Comparative Analysis

```
# Free vs Paid App Analysis
free_vs_paid = df.groupby('Type')['Rating'].mean()
# Category-wise Average Ratings
category_avg_rating = df.groupby('Category')['Rating'].mean()
# Price vs Reviews
price_vs_reviews = df.groupby('Price')['Reviews'].mean()
# Size vs Installs
size_vs_installs = df.groupby('Size')['Installs'].mean()
# Free vs Paid App Size
free_vs_paid_size = df.groupby('Type')['Size'].mean()
# Print outputs
print("Free vs Paid Average Rating:\n", free_vs_paid)
print("\nCategory-wise Average Ratings:\n", category_avg_rating)
print("\nPrice vs Reviews:\n", price_vs_reviews)
print("\nSize vs Installs:\n", size_vs_installs)
print("\nFree vs Paid App Size:\n", free_vs_paid_size)
```

4.6 Correlation

```
# Rating vs Installs Correlation
rating_vs_installs_corr = df['Rating'].corr(df['Installs'])
# Price vs Rating Correlation
price_vs_rating_corr = df['Price'].corr(df['Rating'])
# Size vs Reviews Correlation
size_vs_reviews_corr = df['Size'].corr(df['Reviews'])
# Print outputs
print("Rating vs Installs Correlation: ", rating_vs_installs_corr)
```

```
print("Price vs Rating Correlation: ", price_vs_rating_corr)
print("Size vs Reviews Correlation: ", size_vs_reviews_corr)
```

4.7 Rating and filtering

```
# Top Apps by Reviews
top_apps_by_reviews = df.sort_values(by='Reviews', ascending=False).head(10)
# Top Apps by Installs
top_apps_by_installs = df.sort_values(by='Installs', ascending=False).head(10)
# Paid Apps Distribution
paid_apps_distribution = df[df['Type'] == 'Paid']['Category'].value_counts()
# Print outputs
print("Top Apps by Reviews:\n", top_apps_by_reviews[['App', 'Reviews']])
print("\nTop Apps by Installs:\n", top_apps_by_installs[['App', 'Installs']])
print("\nPaid Apps Distribution:\n", paid_apps_distribution)
```

4.8 Machine Learning code

```
import numpy as np
import matplotlib.pyplot as plt
#Multiple Linear Regression
from sklearn.linear_model import LinearRegression
#Decision Tree
from sklearn.tree import DecisionTreeRegressor
#Random Forest
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, classification_report,
confusion_matrix
# Convert continuous target 'Installs' into categorical for classification metrics
y_train_class = np.where(y_train > np.median(y_train), 1, 0)
y_test_class = np.where(y_test > np.median(y_test), 1, 0)
# Function to train and evaluate models
def train_and_evaluate(model, X_train, y_train, X_test, y_test, name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    # Regression Metrics
```

```
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
    print(f"\n{name} Performance:")
    print(f"RMSE: {rmse:.2f}")
    print(f"R² Score: {r2:.2f}")
    # Convert predictions to categorical for classification metrics
    y_pred_class = np.where(y_pred > np.median(y_test), 1, 0)
    # Classification Report
    print("\nClassification Report:")
    print(classification_report(y_test_class, y_pred_class))
    return y_pred
# Train Models
linear_model = LinearRegression()
tree_model = DecisionTreeRegressor(random_state=42)
forest_model = RandomForestRegressor(n_estimators=100, random_state=42)
y_pred_linear = train_and_evaluate(linear_model, X_train_scaled, y_train, X_test_scaled, y_test, "Linear Regression")
y_pred_tree = train_and_evaluate(tree_model, X_train_scaled, y_train, X_test_scaled, y_test, "Decision Tree Regression")
y_pred_forest = train_and_evaluate(forest_model, X_train_scaled, y_train, X_test_scaled, y_test, "Random Forest Regression")
# Plot Actual vs Predicted
plt.figure(figsize=(12, 5))
plt.scatter(y_test, y_pred_linear, color='blue', label='Linear Regression', alpha=0.5)
plt.scatter(y_test, y_pred_tree, color='green', label='Decision Tree', alpha=0.5)
plt.scatter(y_test, y_pred_forest, color='red', label='Random Forest', alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='black')  # 45-degree line
plt.xlabel("Actual Installs")
plt.ylabel("Predicted Installs")
plt.legend()
```

```
plt.title("Actual vs Predicted Installs")
plt.show()
```

4.9 Cross Validation

```
from sklearn.model_selection import cross_val_score
import numpy as np
# Function to evaluate models using RMSE, R², and Cross-validation
def evaluate_model(model, X_train, y_train, X_test, y_test, model_name):
 model.fit(X_train, y_train)
y_pred = model.predict(X_test)
 # Compute RMSE and R² Score
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
 r2 = r2_score(y_test, y_pred)
# Perform Cross-validation (5 folds)
cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='r2')
mean_cv_r2 = np.mean(cv_scores)
print(f"\nModel: {model_name}")
 print(f"RMSE: {rmse:.2f}")
print(f"R² Score: {r2:.2f}")
 print(f"Cross-validation R²: {mean_cv_r2:.2f}")
return rmse, r2, mean_cv_r2
# Evaluate all models
metrics = []
models = {
    "Linear Regression": linear_model,
    "Decision Tree Regression": tree_model,
    "Random Forest Regression": forest_model
}
for name, model in models.items():
    rmse, r2, cv_r2 = evaluate_model(model, X_train_scaled, y_train, X_test_scaled, y_test, name)
```

```
    metrics.append([name, rmse, r2, cv_r2])
# Convert results to a DataFrame for better visualization
metrics_df = pd.DataFrame(metrics, columns=["Model", "RMSE", "R² Score", "Cross-validation R²"])
print("\nModel Performance Comparison:")
print(metrics_df)
# Plot Feature Importance for Decision Tree & Random Forest
plt.figure(figsize=(10, 5))
for name, model in [("Decision Tree", tree_model), ("Random Forest", forest_model)]:
    if hasattr(model, "feature_importances_"):
        plt.barh(X.columns, model.feature_importances_, label=name, alpha=0.7)
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Feature Importance of Models")
plt.legend()
plt.show()
```

CHAPTER 5
DATA ANALYSIS
&
INTERPRETATION

CHAPTER 5
DATA ANALYSIS & INTERPRETATION

5.1 Summary of data

Statistic	Rating	Size	Installs	Price
Count	7418	7418	7418	7418
Mean	4.171	23306.7	78,30,035	1.117
Std. Dev.	0.55	24002.1	4,63,22,330	17.72
Min	1	8.5	1	0
25%	4	5222.4	10,000	0
50%	4.3	14336	1,00,000	0
75%	4.5	33792	10,00,000	0
Max	5	102400	1,00,00,00,000	400

Interpretation:

The dataset consists of 7,418 app records with an average rating of 4.17 and a size of around 23MB. Installations vary widely, from 1 to 10 billion, with a highly skewed distribution. Most apps are free (median price = $0), but prices go up to $400.

Content Rating	Count
Everyone	5952
Teen	832
Mature 17+	332
Everyone 10+	299
Adults only 18+	2
Unrated	1

Interpretation:

The majority of apps (80%) are rated for "Everyone," making them broadly accessible. Teen-rated apps make up about 11%, while "Mature 17+" and "Everyone 10+" have smaller shares. "Adults only 18+" and "Unrated" apps are extremely rare.

Genre	Count
Tools	633
Entertainment	428
Education	404
Action	318
Personalization	277
...	...
Card;Brain Games	1
Lifestyle;Pretend Play	1
Education;Brain Games	1
Comics;Creativity	1
Strategy;Creativity	1

Interpretation:

The most common genres are Tools (utility apps), Entertainment, and Education, reflecting both practical and leisure-oriented usage. Action games also have a strong presence. Some niche sub-genres, like "Card;Brain Games" and "Strategy;Creativity," appear only once, indicating rare or specialized app categories.

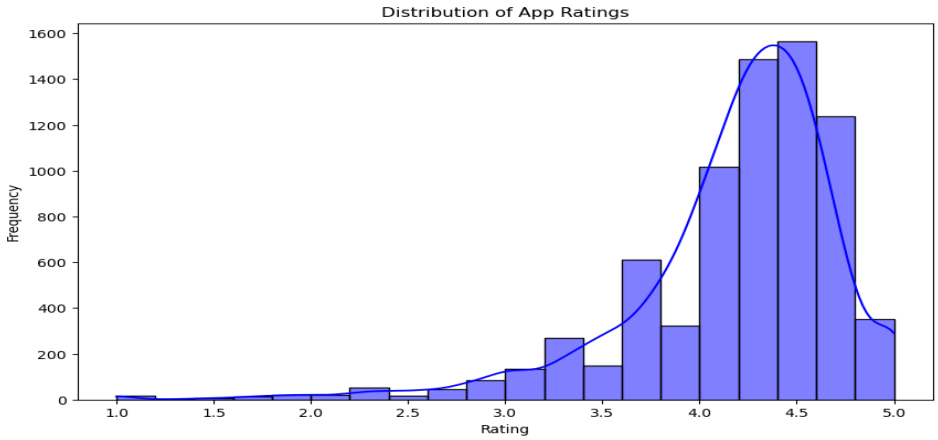
Statistic	Reviews
Count	7418
Unique	4669
Top	2
Frequency	81

Interpretation:

The dataset contains **7,418** reviews in total, with **4,669** unique values. The most frequently occurring review is **"2"**, appearing **81 times**. This suggests that some reviews are repeated, and "2" might be a common rating or sentiment.

5.2 Data Visualisation

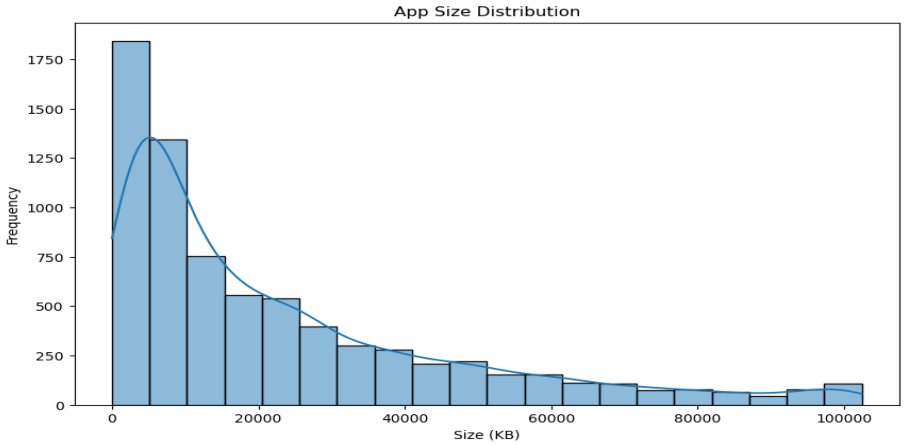
Histogram:



Interpretation:

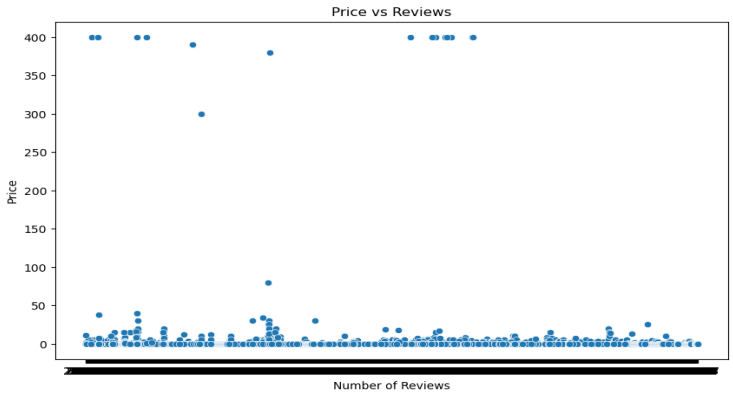
The distribution of app ratings is left-skewed, with most ratings concentrated between 4.0 and 4.8, indicating that users generally rate apps favorably. Very few apps have ratings below 3.0, suggesting that poorly rated apps are uncommon. This trend highlights the importance of quality and user satisfaction, as higher-rated apps tend to be more successful and widely downloaded.

Intrepretation:



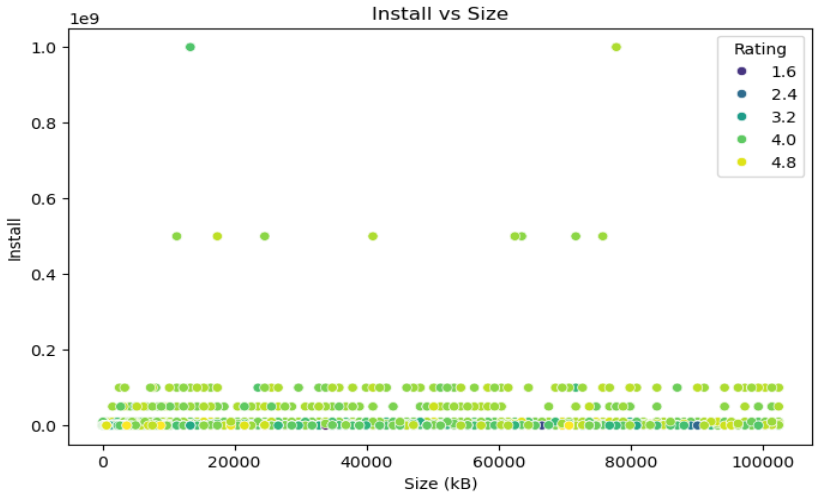
The distribution of app sizes is right-skewed, with most apps being small (0–10,000 KB or 10 MB) and only a few exceeding 50,000 KB (50 MB)**.** The frequency gradually decreases as app size increases, indicating that lightweight apps dominate the dataset, while larger apps are less common. This suggests that **most** developers optimize apps for storage efficiency, with only a few feature-rich or multimedia-heavy applications requiring significantly more space.

Scatter Plot:



Intrepretation:

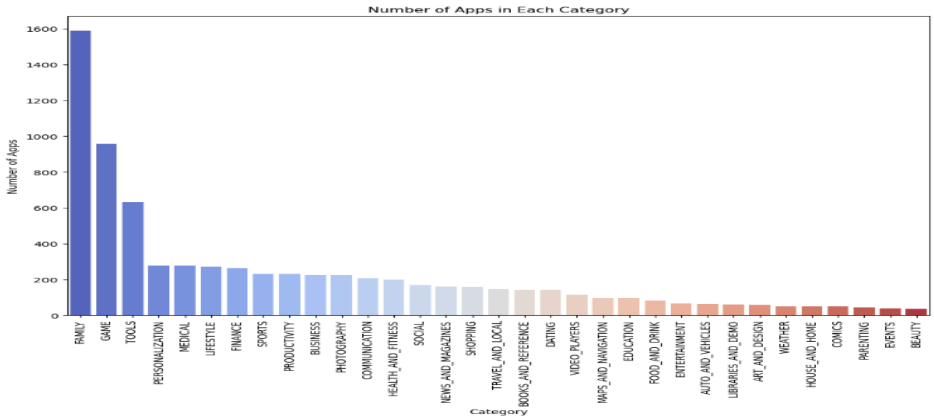
The scatter plot shows the relationship between app price and number of reviews, indicating that most apps are low-priced or free, as seen in the dense cluster near zero. A few apps have higher prices (above $100)**,** but they receive significantly fewer reviews, suggesting that expensive apps may have limited user engagement. This trend implies that users are more likely to download and review affordable or free apps compared to costly ones.



Intrepretation:

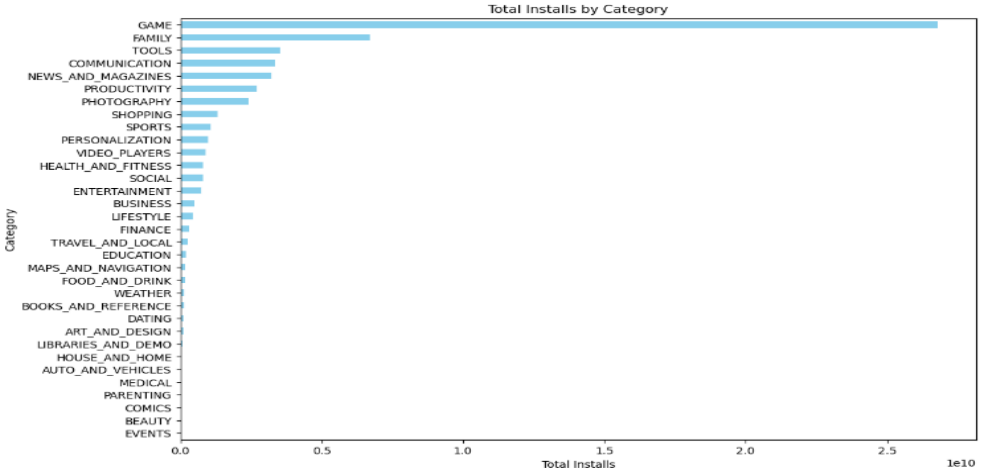
The scatter plot visualizes the relationship between app size (in kB) and the number of installs, with color indicating the app rating. Most apps have relatively small sizes and low install counts, while a few larger apps have significantly higher installs, reaching nearly a billion. The distribution suggests that app size does not strongly impact the number of installs, but higher-rated apps (yellow-green) appear more frequently among the popular ones.

Bar Plot



Intrepretation:

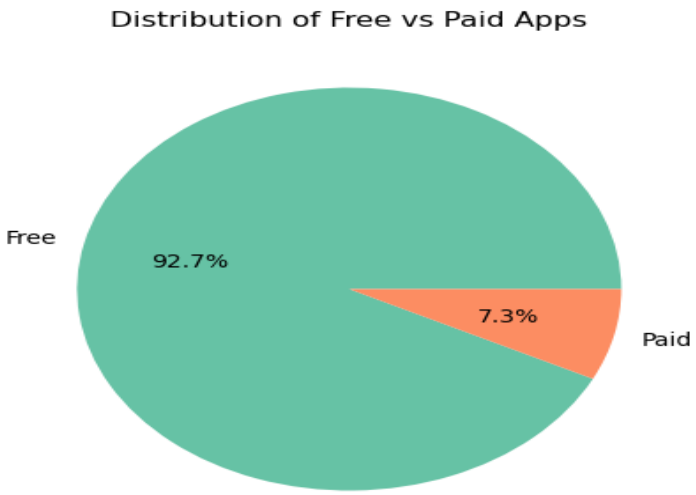
The bar chart displays the number of apps in different categories, with Family and Game categories having the highest number of apps. Other categories like Tools, Personalization, and Medical also have a significant presence. Meanwhile, categories like Beauty, Events, and Parenting have the fewest apps, indicating a lower focus or demand in these areas.



Intrepretation:

The bar chart shows total app installs across different categories, with Games having the highest number of installs by a large margin, followed by Family and Tools. Categories like Communication, News, and Productivity also have significant installs, indicating high user engagement. Meanwhile, categories like Beauty, Comics, and Events have the lowest installs, suggesting a smaller user base.

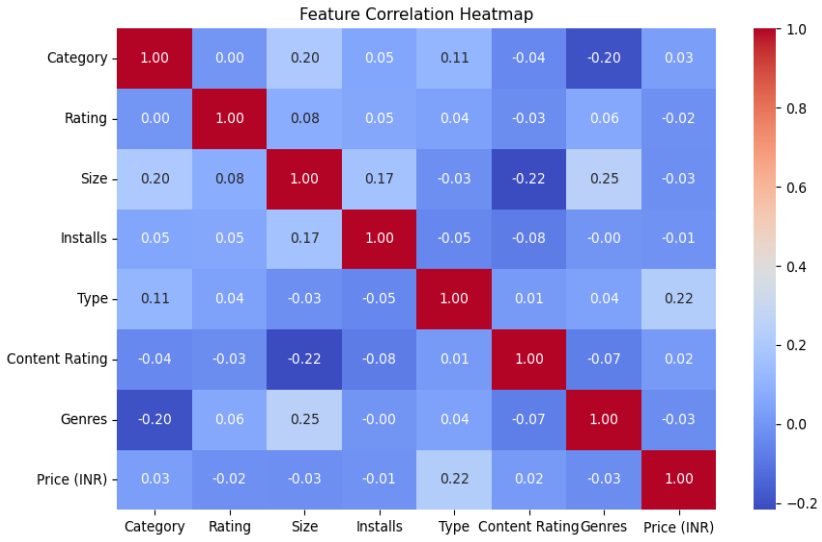
Pie Chart:



Intrepretation:

The pie chart illustrates the distribution of **free vs. paid apps**, showing that **92.7%** of the apps are free, while only **7.3%** are paid. This indicates that the majority of apps follow a free-to-use model, likely relying on ads or in-app purchases for revenue, while a smaller fraction operates on a direct purchase model.

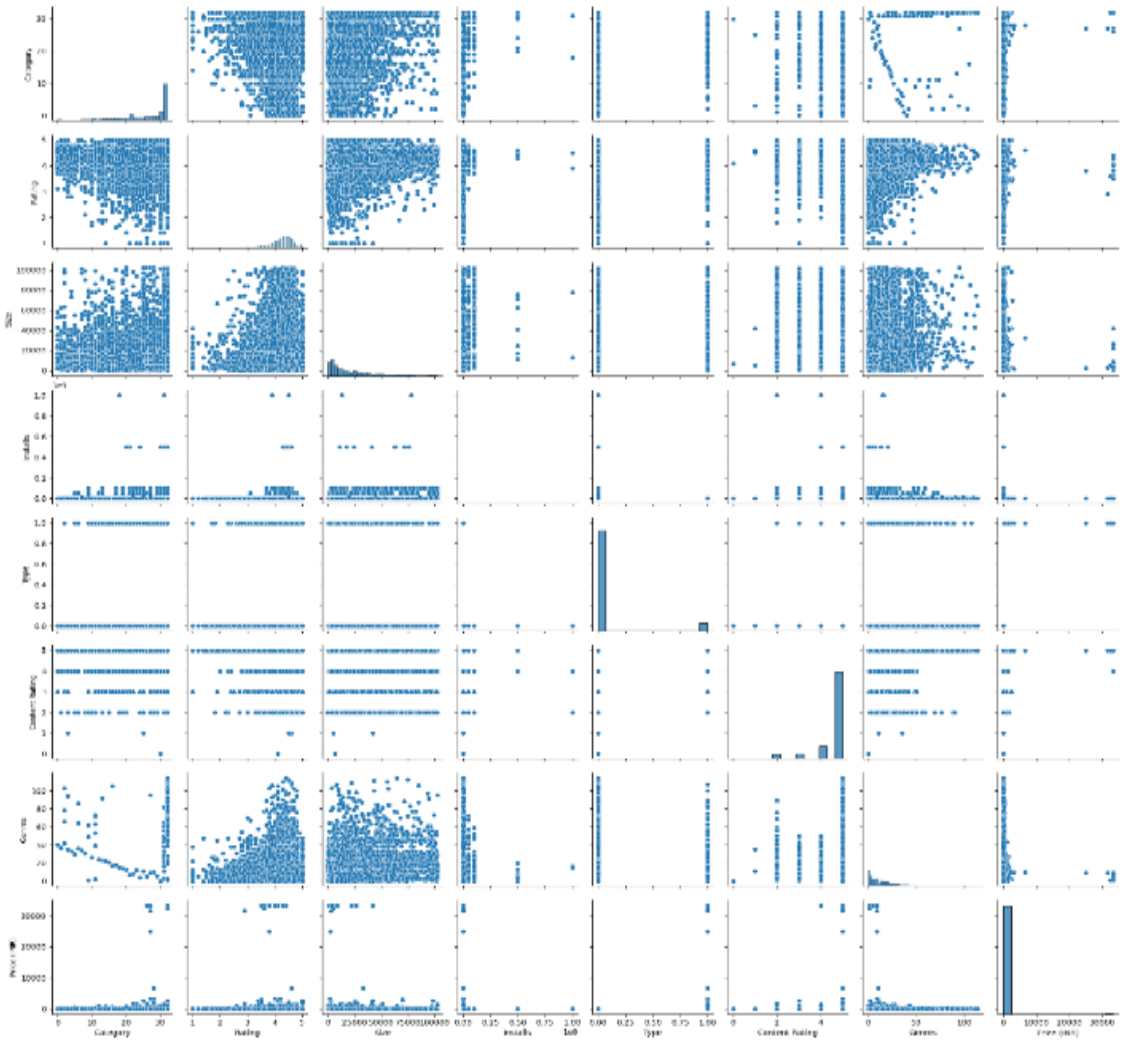
Correlation Matrix:



Intrepretation:

The heatmap shows the correlation between different app attributes**.** Size has a slight positive correlation with Installs and Genres, suggesting that larger apps tend to be downloaded more. Price has a weak correlation with Type and Content Rating, indicating that paid apps may vary slightly based on content suitability. Overall, the correlations are relatively weak, implying that no single factor strongly influences app success.

Pair Plot:



Intrepretation:

The scatterplot matrix provides a visual representation of relationships between multiple variables. Some plots show positive or negative trends, indicating possible correlations, while others appear more randomly distributed, suggesting no strong relationship. The presence of clustered points and outliers highlights potential patterns and anomalies in the dataset.

5.3 Comparative Analysis

- **Free vs Paid Average Rating:**

  Free 4.165357

  Paid 4.247523

- **Category-wise Average Ratings:**

  GAME 4.265693

  FAMILY 4.189308

  COMMUNICATION 4.097573

  ART_AND_DESIGN 4.381034

  EVENTS 4.478947

- **Free vs Paid Average App Size (in KB):**

  Type

  Free 23484.251928

  3Paid 21068.032661

Interpretation:

Free vs. Paid Average Rating: Paid apps have a slightly higher average rating (4.2475) compared to free apps (4.1654). This suggests that users tend to rate paid apps slightly better, possibly due to better quality, fewer ads, or enhanced user experience. However, the difference is not very large.

Category-wise Average Ratings: The highest-rated category is **Events (4.4789)**, followed by **Art & Design (4.3810)** and **Games (4.2657)**. This indicates that apps in the **Events and Art & Design** categories generally receive higher user satisfaction. This could be due to their specialized nature or lower user expectations compared to more competitive categories like Communication. The **Communication category has the lowest rating (4.0976)**, possibly due to issues with functionality, frequent updates, or user frustrations with ads and in-app purchases

Free vs. Paid Average App Size: Free apps tend to be **larger in size (23,484 KB) compared to paid apps (21,068 KB)**. This might be because free apps include additional assets such as advertisements, third-party integrations, or freemium features that contribute to a larger file size. Paid apps, being smaller, could be more optimized and focused on delivering premium features without the unrelated stuffs.

5.4 Rank Analysis

Top Paid Apps Distribution by Categories:

- FAMILY – 146
- GAME – 75
- PERSONALIZATION – 62
- TOOLS – 55
- MEDICAL – 55
- SPORTS – 20

Top Apps by Reviews:

- Clash of Clans – 44893888
- Subway Surfers - 27725352
- Clash Royale - 23136735

Top Apps by Installs:

- Subway Surfers – 1000000000+
- Google News – 1000000000+
- Dropbox – 500000000+
- Candy Crush Saga - 500000000+

Interpretation:

Top Paid Apps Distribution by Categories:

The FAMILY category dominates (146 apps), meaning many paid apps cater to family-friendly content such as educational tools, kids' entertainment, or parental control apps. Games (75 apps) hold the second spot, which is expected since premium games often offer ad-free experiences and exclusive features. Personalization (62 apps) indicates a strong demand for custom themes, widgets, and wallpapers in the paid segment. Tools and Medical categories (55 each) show that utility and healthcare-related apps are also valued in the paid app market, possibly because they provide critical functionalities without ads or interruptions.

Top Apps by Reviews:

Clash of Clans (44.89M reviews) leads in user engagement, suggesting its strong, active community and high retention rate. Subway Surfers (27.72M) and Clash Royale (23.13M) also have significant engagement, reflecting their popularity and frequent updates. All three are highly competitive, strategy-based, or endless-runner games, which attract massive player bases and generate strong user interactions.

Top Apps by Installs:

Subway Surfers and Google News both exceed 1 billion installs, making them some of the most widely used apps globally. Dropbox and Candy Crush Saga (500M+ installs each) highlight the success of cloud storage and casual gaming in attracting massive user bases. Notably, productivity (Dropbox) and news apps (Google News) also compete with gaming apps in terms of downloads, showing that utility apps can achieve widespread adoption alongside entertainment apps.

5.5 Machine Learning Analysis

Multiple Linear Regression :

RMSE: 43576028.61
R^2 Score: 0.48

Classification Report:

	precision	recall	f1-score	support
0	0.95	0.07	0.13	798
1	0.48	1.00	0.65	686
accuracy			0.50	1484
macro avg	0.71	0.53	0.39	1484
weighted avg	0.73	0.50	0.37	1484

Decision Tree :

RMSE: 12098944.74
R^2 Score: 0.96

Classification Report:

	precision	recall	f1-score	support
0	0.94	0.93	0.94	798
1	0.92	0.93	0.93	686
accuracy			0.93	1484
macro avg	0.93	0.93	0.93	1484
weighted avg	0.93	0.93	0.93	1484

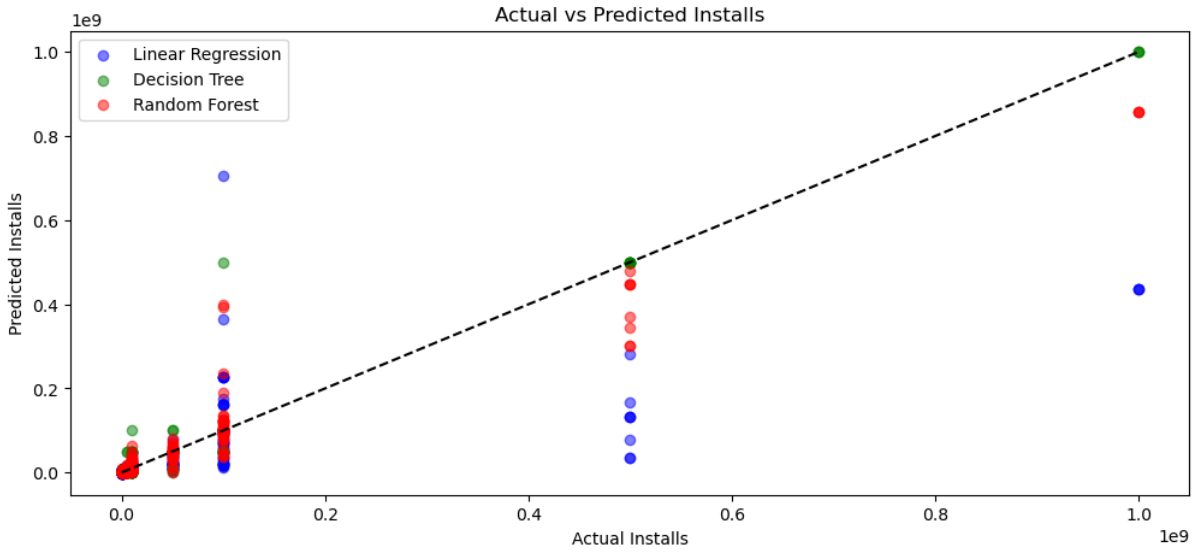
Random Forest Regression:

RMSE: 17215813.76
R^2 Score: 0.92

Classification Report:

	precision	recall	f1-score	support

0	0.99	0.79	0.88	798
1	0.80	0.99	0.89	686
accuracy			0.88	1484
macro avg	0.90	0.89	0.88	1484
weighted avg	0.90	0.88	0.88	1484



Interpretation:

The scatter plot visualizes the performance of Multiple Linear Regression, Decision Tree, and Random Forest models in predicting app installs. The dashed diagonal line represents perfect predictions, where actual and predicted values are equal. The Decision Tree and Random Forest models perform better, with predictions closely aligned to the line, whereas Linear Regression shows greater deviation, especially for high install values. This suggests that non-linear models like Decision Tree and Random Forest capture patterns more effectively than Linear Regression.

5.6 Cross validation

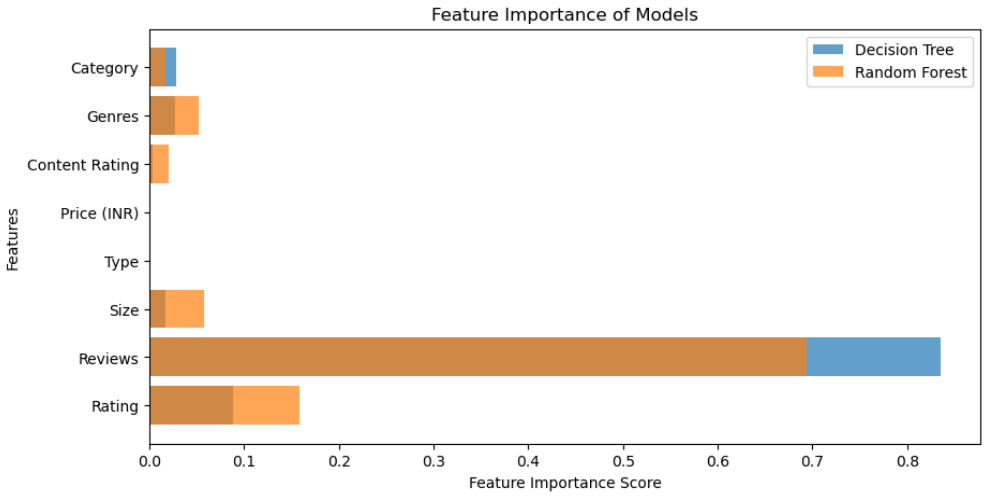
Model: Multiple Linear Regression
RMSE: 43576028.61
R^2 Score: 0.48
Cross-validation R^2: 0.31

Model: Decision Tree Regression
RMSE: 12098944.74
R^2 Score: 0.96
Cross-validation R^2: 0.61

Model: Random Forest Regression
RMSE: 17215813.76
R^2 Score: 0.92
Cross-validation R^2: 0.7

Model Performance Comparison:

	Model	RMSE	R^2 Score	Cross-validation R^2
0	Linear Regression	4.357603e+07	0.477912	0.310479
1	Decision Tree Regression	1.209894e+07	0.959752	0.609703
2	Random Forest Regression	1.721581e+07	0.918510	0.710582



Interpretation

The bar chart compares the feature importance scores for Decision Tree and Random Forest models. Reviews have the highest importance, indicating their strong influence on predictions. Rating and Size also contribute significantly, whereas Category, Genres, and Content Rating have lower importance. The Random Forest model assigns more weight to Rating compared to the Decision Tree. This suggests that user engagement (reviews and ratings) plays a crucial role in app performance prediction.

Sample Prediction Results

Actual Instal	Linear Regression Pred	Decision Tree Predic	Random Forest Predi
1,000,000	5,190,866	1,000,000	1,240,000
1,000	2,608,702	1,000	564
5,000	2,919,826	10,000	14,350
10,000,000	6,425,457	10,000,000	9,190,000
10,000,000	9,349,279	10,000,000	10,700,000

Interpretation:

The table compares actual app installs with predictions from Linear Regression, Decision Tree, and Random Forest models. Linear Regression significantly overestimates installs for smaller values but is closer for larger ones. The Decision Tree model provides exact matches for higher values but struggles with smaller ones. Random Forest offers more balanced predictions but still has notable deviations, especially at lower install counts. Overall, each model has strengths and weaknesses depending on the data range.

CHAPTER 6 CONCLUSION & FUTURE SCOPE

CHAPTER 6

CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION:

Based on the evaluation metrics, the Decision Tree model performs the best among the three models tested. It has the lowest RMSE (12,098,944.74) and the highest R^2 score (0.96**)**, indicating a strong predictive performance. Additionally, its classification report shows high precision, recall, and F1-scores, suggesting that it effectively distinguishes between classes.

The Random Forest Regression model also performs well, with an R^2 score of 0.92 and lower RMSE (17,215,813.76) compared to Multiple Linear Regression. However, its classification performance is slightly lower than the Decision Tree, especially in recall and F1-score.

On the other hand, Multiple Linear Regression performs the worst, with a high RMSE (43,576,028.61) and a low R^2 score (0.48), indicating that it struggles to capture the patterns in the data effectively.

6.2 FUTURE SCOPE:

Given your dataset, which predicts app installs based on features like ratings, reviews, size, and price, ensemble methods can enhance predictive performance in several ways:

- **Hybrid Ensembles for Improved Accuracy**: Combining models like Gradient Boosting, Random Forest, and Neural Networks can lead to more robust predictions by capturing different aspects of the data, reducing overfitting and variance.
- **AutoML-Driven Optimization**: Tools like TPOT and H2O AutoML can automate model selection, hyperparameter tuning, and ensemble formation, improving efficiency while ensuring the best model combinations.
- **Deep Learning Ensembles**: Using ensembles of DNNs with TensorFlow/Keras, possibly with attention mechanisms or transformers, can improve accuracy in handling complex patterns in app install trends.
- **Real-Time Predictive Analytics**: Deploying ensemble models in real-time app store analytics can help businesses dynamically adjust marketing strategies based on install predictions.
- **Personalized App Recommendations**: Using ensemble learning in combination with reinforcement learning could help app stores provide personalized recommendations based on user preferences and install trends.
- **Scalability for Large Datasets**: As app data grows, distributed ensemble learning (using frameworks like Apache Spark ML or TensorFlow Extended) can improve scalability and computational efficiency.
- **Explainable AI for Business Decisions**: Implementing explainable ensemble models (XAI) can help businesses understand why certain apps receive more installs, aiding in better decision-making.

BIBLIOGRAPHY:

1. Data Source: www.kaggle.com
2. pythonspot.com
3. diveintopython3.net
4. python.swaroopch.com
5. learnpython.org
6. Michal Jaworski, Tarek Zindé (2021):" Expert Python Programming", 4th Edition, Packt Publishing Limited.
7. Manaranjan Pradhan, U Dinesh Kumar (2019):" Machine Learning Using Python.
8. Python: Data Analytics and Visualization-2017-Ashish kumar, Kirthi Raman, PACKT publications.
9. Wes McKinney, (2011), "Python for Data Analysis", O'Reilly (Third Edition).

Reviews:

1. Prediction Analysis on Google Play Store, by Qredness, in Kaggle.com.
2. EDA on Google Play Store App, by Peeyush Kant Misra, in Kaggle.com.
3. Analysis of Total Installs by App Category, by Yaseen Abdul Ghany, in Kaggle.com.
4. Exploratory Analysis on Google play store Dataset, by Claudio, in Kaggle.com.
5. LLM Gemma | Engineering prompt sentiment analysis, by Rafael Gallo, in Kaggle.com.
6. Basic Google Play Store App analytics, by Gautham Prakash, in gauthamp10.github.io.
7. Data visualization and Sentiment Analysis, by Zweli Khumalo, in datacamp.com.
8. Data Visualization and Distribution Understanding of Google play Store App Data, by Ivana Lucia Kharisma, datacamp.com.
9. Google Play Store Dataset: Visualizing using Correlation Matrix and Kernel Density Estimates for Histograms, by Kabila MD Musa, in Medium.com.
10. Data Visualization on Google Play Store, by Arthur Muratov, in Medium.com.