

DATA SCIENCE/MACHINE LEARNING

CODING BLOCKS

Submitted in partial fulfillment of the requirements for the award of a degree of

BTECH – CSE

(Machine Learning)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB



From 06/09/2024 to 07/31/2024

SUBMITTED BY

Name of student: Munukutla Lakshmi Lavanya

Registration Number: 12223517

Annexure-II: Student Declaration

To whom so ever it may concern

I, Munukutla Lakshmi Lavanya, Reg No:12223517 at this moment declare that the work done by me on “**Data Science/Machine Learning**” from **June 2024** to **July 2024**, is a record of original work for the partial fulfillment of the requirements for the award of the degree, **Data Science**

Name of the Student: Munukutla Lakshmi Lavanya

Registration Number: 12223517

Dated:30|08|2024

TABLE OF CONTENTS

Cover Page.....	1
Student Declaration.....	2
Acknowledgement.....	4
Certificate.....	5
Chapter 1: INTRODUCTION OF THE PROJECT.....	6
1.1 Objectives of the work undertaken	
1.2 Scope of the Work	
1.3 Importance and Applicability	
1.4 Applications of Machine Learning	
Chapter 2: INTRODUCTION OF THE COMPANY/WORK.....	8
2.1 Company's Vision and Mission	
2.2 Origin and growth of the company	
2.3 Various departments and their functions	
2.4 Relavance	
Chapter 3: Brief description of the work done.....	10
3.1 Position of Internship	
3.2 Data Collection	
3.3 Feature Engineering	
3.4 Machine Learning Algorithms	
3.5 Model Evaluation	
Chapter 4: Project	
4.1 Learning Outcomes	
4.2 Data Analysis	
Conclusion.....	27
References.....	28

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Coding Blocks for providing an exceptional learning experience through their comprehensive program. The quality of the content and the depth of the material have significantly contributed to my understanding and skills in the field. I extend my sincere thanks to the instructors and contributors who designed and delivered the course, making complex topics accessible and engaging. Their dedication and expertise were invaluable in enhancing my learning journey. I am also grateful to my university and professors for their continuous support and encouragement throughout my studies. Their guidance has been crucial in motivating me to apply the knowledge gained from this program in real-world scenarios. Lastly, I would like to thank my family and friends for their unwavering support and encouragement, which kept me motivated throughout the duration of this program.

SUMMER TRAINING CERTIFICATE BY CODING BLOCKS



CERTIFICATE

of completion

Proudly Presented To:

M Lakshmi Lavanya

in recognition for successfully completing **Summer Internship - Data Science.**



Jun 2024-Jul 2024

Date

Varun Kohli

Varun Kohli
Co-founder & CEO

The certificate can be verified at
<https://online.codingblocks.com/app/certificates/CBOL-302114-e5a53975>

Chapter 1: INTRODUCTION OF THE PROJECT UNDERTAKEN

1.1 Objectives of the work undertaken

The main objectives of this work are to build a strong understanding of Python programming, and data scrapping. It began with the basics of maths like linear algebra, and probabilities, which are important for understanding how machine learning works. Along with them, I have learned machine learning algorithms such as Linear Regression, Decision Tree, SVM, Logistic Regression, and KNN. The final part focuses on deep learning techniques like CNN, and GAN.

- Understanding Machine Learning (ML): The aim is to grasp the basics of ML, including its types (supervised, unsupervised, and reinforcement learning) and their applications. ML involves creating algorithms that allow computers to learn from and make predictions or decisions based on data.
- Exploring Data Science: To learn how data science integrates statistical analysis, data manipulation, and machine learning to uncover insights from data. Data science combines techniques from computer science and statistics to interpret and analyze complex datasets.
- Application of Techniques: To understand how various ML and data science techniques are applied to solve real-world problems. This includes practical applications in different industries and the use of specific tools and methodologies for

data analysis.

1.2 Scope of the Work

- **Machine Learning: The scope includes:**
- Supervised Learning: Algorithms that learn from labeled data to make predictions or classifications. Examples include linear regression, logistic regression, and support vector machines.
- Unsupervised Learning: Algorithms that identify patterns in unlabeled data.

Examples include clustering algorithms like K-means and dimensionality reduction techniques like Principal Component Analysis (PCA).

- Reinforcement Learning: Algorithms that learn by interacting with an environment and receiving feedback. Examples include Q-learning and policy gradient methods.

Data Science: The scope encompasses:

- Data Collection: Gathering data from various sources such as databases, APIs, and web scraping.
- Data Cleaning: Preprocessing data to handle missing values, outliers, and inconsistencies.
- Exploratory Data Analysis (EDA): Using statistical techniques and visualizations to understand data distributions and relationships.

- **Data Visualization:** Creating graphs and charts to represent data insights effectively.
- **Integration:** How ML models are integrated into data pipelines and systems for practical use. This includes deploying models in production environments and using them for real-time data analysis and decision-making.

1.3 Importance

Machine learning is being used in more and more areas, making it an essential tool today. As technology advances, machine learning will continue to play a bigger role in driving new ideas and improving things, impacting almost every part of our lives. The skills learned are directly applicable to solving real-world problems that come across various industries, retail, and healthcare, where decision making is crucial. It also helps in career development in data science and Machine Learning. Its ability to process vast amounts of data quickly and accurately, uncovering insights that would be impossible for humans to detect manually.

1.4 Applications of Machine Learning

- 1. Healthcare:** Beyond diagnosing diseases and predicting outcomes, machine learning is used in medical imaging and treatment plans.
- 2. Finance:** Machine learning analyse transaction patterns in real-time to identify suspicious activities.
- 3. Manufacturing:** In manufacturing, machine learning is used for predictive maintenance, where it can predict equipment failures before they occur.

4. Transportation and Logistics: Machine learning is at the heart of autonomous vehicles, enabling them to navigate and make decisions in real time.

5. Cybersecurity: In cybersecurity, machine learning helps in detecting and responding to threats faster than traditional methods. It can identify unusual patterns



Chapter 2: INTRODUCTION OF THE COMPANY/WORK

2.1 Company's Vision and Mission

Vision: Coding Blocks aims to empower people by teaching them coding skills that are essential in today's tech-driven world. They want to make coding education accessible and practical, helping students and professionals build successful careers in technology.

Mission: The mission of Coding Blocks is to provide high-quality, hands-on coding education that bridges the gap between theoretical learning and real-world application. They strive to create a learning environment where students can gain the skills they need to excel in the tech industry.

2.2 Origin and growth of the company

Origin: Coding Blocks was founded in 2014 by Manmohan Gupta, Arnav Gupta, and Bansal Varun. They started the company to offer practical coding education that goes beyond what is typically taught in traditional classrooms.

Growth: The company started with in-person classes and quickly expanded to offer online courses. Over the years, Coding Blocks has grown into a leading platform for coding education in India, known for its comprehensive courses and strong community of learners.

2.3 Various departments and their functions

1. Education Department: Responsible for designing and delivering the courses, including curriculum development and hiring instructors.
2. Technology Department: Manages the online learning platform, ensuring it runs smoothly and is user-friendly. They also develop any software tools used in the courses.
3. Marketing and Sales: Focuses on promoting the courses, reaching out to potential students, and handling enrolments.
4. Customer Support: Provides help to students with any issues they might face, whether technical or related to the course content.
5. Human Resources (HR): Manages recruitment, employee relations, and the overall work environment at Coding Blocks.
6. Partnerships and Collaborations: Works on building relationships with other educational institutions and companies for internships, placements, and special programs.

2.4 Relavance:

Career Opportunities: Expertise in ML and data science opens up numerous career opportunities.

Roles include:

1. Data Scientist: Analyzes and interprets complex data to provide actionable insights.
2. Machine Learning Engineer: Designs and implements ML models and algorithms.

3.4 Business Analyst: Uses data analysis to support business decision-making.

4. Research and Development: The knowledge gained in ML and data science provides a foundation for pursuing research in advanced topics such as deep learning, reinforcement learning, and artificial intelligence. This research contributes to technological advancements and innovation in the field.

Chapter 3: Brief description of the work done

3.1 Position of Internship and Roles

As a part of my summer internship, I'm a trainee in a huge coding blocking company and I took my summer training from June to July. I have done assignments using machine learning algorithms. The team supported me during the training to finish my project.

3.2 Data Collection, Cleaning, and Preprocessing

Data Collection

Data collection is a crucial first step in any data science or machine learning project. Accurate and relevant data forms the foundation upon which models are built and insights are derived.

Without quality data, the results of any analysis or model will be unreliable.

- **Databases:** Data can be collected from relational databases (e.g., MySQL, PostgreSQL) using SQL queries. These databases store structured data and are commonly used in business applications.
- **APIs (Application Programming Interfaces):** Many services provide APIs to access data programmatically. For instance, Twitter and Google Maps offer APIs that allow for real-time data retrieval.
- **Web Scraping:** This involves extracting data from websites using tools like BeautifulSoup or Scrapy in Python. Web scraping is useful for gathering data from sites that do not provide APIs.
- **Surveys and Forms:** Data can also be collected through surveys and forms, which are useful for gathering primary data from respondents. Tools like Google Forms and SurveyMonkey facilitate this process.

Techniques for Data Collection

- **Manual Collection:** Involves manually entering data or collecting data through direct observation or interviews. This method is often used when automated methods are not feasible.

- **Automated Collection:** Uses scripts or software to collect data at scale. This includes automated data extraction from APIs or web scraping, which can handle large volumes of data efficiently.

- **Real-Time Data Collection:** Involves collecting data in real-time from sources like sensors, IoT devices, or streaming services. Real-time data is crucial for applications requiring immediate analysis, such as fraud detection or live monitoring.

Data Cleaning

Data cleaning is essential for ensuring the accuracy and quality of data. Raw data often contains errors, inconsistencies, and missing values, which can lead to misleading or incorrect results if not addressed properly.

Techniques for Data Cleaning

- **Handling Missing Values:** Missing data can occur for various reasons, such as incomplete records or errors in data collection. Common strategies include:

- **Imputation:** Filling in missing values with statistical estimates (e.g., mean, median) or using algorithms to predict missing values.

- **Deletion:** Removing records with missing values, although this can lead to loss of valuable information.

- **Flagging:** Creating a new feature to indicate missing values, allowing models to handle them separately.

- **Outlier Detection and Handling:** Outliers are extreme values that deviate significantly from other observations. Techniques include:

- **Statistical Methods:** Identifying outliers using statistical measures (e.g., Z-scores, IQR).

- **Visualization:** Using plots (e.g., box plots) to detect outliers visually.

- **Transformation:** Applying transformations (e.g., logarithmic) to reduce the impact of outliers.

- **Data Consistency:** Ensuring that data is uniform and consistent across the dataset. This involves:

- **Standardization:** Converting data to a standard format (e.g., date formats, text capitalization).
- **Validation:** Checking for and correcting inconsistencies in data entries.
- **Data Integration:** Combining data from multiple sources into a cohesive dataset. This may involve:
 - **Merging:** Joining datasets based on common attributes (e.g., merging customer data from different departments).
 - **Aggregation:** Summarizing data to provide a comprehensive view (e.g., aggregating sales data by region).

Data Preprocessing

Data preprocessing prepares data for analysis by transforming it into a format suitable for machine learning models. Proper preprocessing improves model performance and ensures that data is ready for analysis.

3.3 Feature Engineering

Definition: Creating new features from raw data to enhance the model's performance.

Feature engineering involves transforming existing data into meaningful attributes.

Data Transformation

- **Normalization and Scaling:** Adjusting the range of features to a standard scale. Techniques include:

- **Min-Max Scaling:** Rescaling features to a range between 0 and 1.
- **Standardization:** Adjusting features to have a mean of 0 and a standard deviation of 1.

- **Data Encoding:** Converting categorical data into numerical format. Techniques include:

- **One-Hot Encoding:** Creating binary columns for each category.
- **Label Encoding:** Assigning unique integers to categories.
- **Dimensionality Reduction:** Reducing the number of features while preserving essential information. Techniques include:

- Principal Component Analysis (PCA): Transforming features into a lower-dimensional space while retaining most of the variance.
- Feature Selection: Selecting a subset of relevant features based on their importance to the model.

Data Splitting

- Purpose: Dividing data into subsets for training, validation, and testing. This ensures that the model is evaluated on unseen data and helps prevent overfitting.
- Techniques:
 - Training Set: Used to train the model and fit it to the data.
 - Validation Set: Used to tune model parameters and select the best model.
 - Test Set: Used to assess the final performance of the model.

Tools and Libraries

- Python Libraries: Libraries such as Pandas, NumPy, and Scikit-Learn provide functions for data cleaning and preprocessing.
- R Packages: Tools like dplyr and tidyr are used for data manipulation and preparation in R.

3.4 Machine Learning Algorithms and Models

Machine Learning (ML) algorithms are the core components that enable machines to learn from data and make predictions or decisions. Understanding different types of ML algorithms is crucial for selecting the right model for a specific problem.

Classification of Machine Learning Algorithms

- Supervised Learning: Algorithms learn from labeled data to make predictions or classifications.
- Regression: Predicts continuous values.
- Classification: Predicts discrete categories.
- Unsupervised Learning: Algorithms identify patterns in unlabeled data.
- Clustering: Groups similar data points together.

- Dimensionality Reduction: Reduces the number of features while preserving essential information.
- Reinforcement Learning: Algorithms learn by interacting with an environment and receiving feedback. Used in scenarios where learning involves trial and error.

Key Concepts

- Model Training: The process of using data to adjust the parameters of a machine learning model so it can make accurate predictions.
- Overfitting and Underfitting:
 - Overfitting: When a model performs well on training data but poorly on unseen data, usually due to excessive complexity.
 - Underfitting: When a model is too simple to capture the underlying patterns in the data.

Supervised Learning Algorithms

Linear Regression

- Definition: A regression algorithm that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation.
- Applications: Predicting sales, forecasting stock prices, and estimating property values.
- Key Metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

Logistic Regression

- Definition: A classification algorithm used to predict binary outcomes by estimating probabilities using a logistic function.
- Applications: Email spam detection, medical diagnosis, and credit scoring.
- Key Metrics: Accuracy, Precision, Recall, F1 Score, and ROC Curve.

Decision Trees

- Definition: A tree-like model used for both classification and regression, where data is split into branches based on feature values.

- Applications: Customer segmentation, loan approval, and risk assessment.
- Key Metrics: Gini Index, Entropy, and Classification Accuracy.

Random Forest

- Definition: An ensemble learning method that creates multiple decision trees and combines their outputs to improve accuracy and prevent overfitting.
- Applications: Feature selection, classification problems, and regression tasks.
- Key Metrics: Out-of-Bag Error, Feature Importance, and Mean Squared Error.

Support Vector Machines (SVM)

- Definition: A classification algorithm that finds the optimal hyperplane that separates different classes in the feature space.
- Applications: Image classification, text classification, and bioinformatics.
- Key Metrics: Margin of Separation, Support Vectors, and Classification Accuracy.

K-Nearest Neighbors (KNN)

- Definition: A simple classification algorithm that assigns a class to a data point based on the majority class of its k-nearest neighbors.
- Applications: Recommender systems, image recognition, and anomaly detection.
- Key Metrics: Classification Accuracy, Precision, Recall, and F1 Score.

Unsupervised Learning Algorithms

K-Means Clustering

- Definition: A clustering algorithm that partitions data into k distinct clusters based on feature similarity.
- Applications: Market segmentation, image compression, and anomaly detection.
- Key Metrics: Silhouette Score, Inertia, and Cluster Centroids.

Hierarchical Clustering

- Definition: A clustering algorithm that creates a hierarchy of clusters by either merging or splitting them iteratively.
- Applications: Social network analysis, taxonomies, and gene expression analysis.

- Key Metrics: Dendrogram, Cophenetic Correlation Coefficient, and Cluster Validity Index.

Principal Component Analysis (PCA)

- Definition: A dimensionality reduction technique that transforms data into a lower-dimensional space while preserving variance.
- Applications: Data visualization, noise reduction, and feature extraction.
- Key Metrics: Explained Variance Ratio, Principal Components, and Reconstruction Error.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Definition: A technique for dimensionality reduction and visualization, especially useful for high-dimensional data.
- Applications: Visualizing clusters in high-dimensional data, exploring datasets.
- Key Metrics: Perplexity, Variance Explained, and Visualization Quality.

Reinforcement Learning Algorithms

Q-Learning

- Definition: A model-free reinforcement learning algorithm that learns the value of actions in various states to maximize cumulative rewards.
- Applications: Game playing, robotics, and autonomous driving.
- Key Metrics: Q-Value Updates, Policy Improvement, and Cumulative Reward.

Policy Gradient Methods

- Definition: A class of algorithms that optimize the policy directly by updating policy parameters to maximize expected rewards.
- Applications: Robotics control, game strategy optimization, and resource management.
- Key Metrics: Policy Gradient Estimation, Reward Maximization, and Training Stability.

3.5 Model Evaluation and Selection

Cross-Validation

- Definition: A technique for assessing how the results of a statistical analysis generalize to an independent data set by dividing the data into training and testing subsets.
- Applications: Model performance evaluation, parameter tuning, and model comparison.
- Key Metrics: K-Fold Cross-Validation Score, Leave-One-Out Cross-Validation Score, and Model Stability.

Hyperparameter Tuning

- Definition: The process of optimizing model parameters that are not learned from data but are set before training to improve model performance.
- Applications: Model optimization, performance improvement, and achieving better generalization.
- Key Metrics: Grid Search, Random Search, and Bayesian Optimization.

Performance Metrics

- Classification Metrics: Accuracy, Precision, Recall, F1 Score, ROC-AUC.
- Regression Metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared.
- Clustering Metrics: Silhouette Score, Davies-Bouldin Index, Calinski-Harabasz Index

Model Evaluation

Evaluation Metrics

Evaluation metrics are used to assess the performance of a machine learning model. The choice of metric depends on the type of problem (classification, regression, clustering, etc.).

- Classification Metrics:
- Accuracy: The ratio of correctly predicted instances to the total instances.
- Precision: The ratio of true positive predictions to the total predicted positives.
- Recall (Sensitivity): The ratio of true positive predictions to the total actual

positives.

- F1 Score: The harmonic mean of precision and recall, balancing the two metrics.
- ROC Curve and AUC: The Receiver Operating Characteristic curve plots the true positive rate against the false positive rate, while the Area Under the Curve (AUC) measures the overall performance.
- Regression Metrics:
 - Mean Absolute Error (MAE): The average of absolute differences between predicted and actual values.
 - Mean Squared Error (MSE): The average of squared differences between predicted and actual values.
 - R-squared: The proportion of variance in the dependent variable that is predictable from the independent variables.

Clustering Metrics:

- Silhouette Score: Measures how similar an object is to its own cluster compared to other clusters.
- Davies-Bouldin Index: Evaluates the average similarity ratio of each cluster with its most similar cluster.

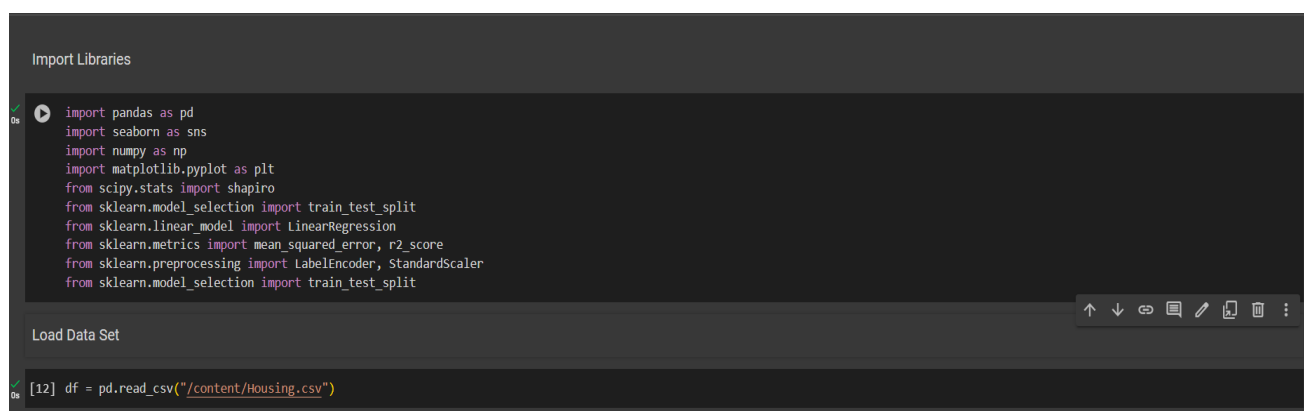
Chapter 4: Project(House Price Prediction)

I have done a project on Predicting House Prices in a given data set and implemented an algorithm (Linear Regression).

- Dataset: The dataset used in this analysis is taken from Kaggle “Housing Prices”. It contains certain information about a particular region, with features like the Number of bathrooms, bedrooms, and more.
- Data Description:

The dataset consists of several features (columns) that describe different aspects of the houses. Here's an overview of the key columns:

- prices: The target variable indicates the sale price of the properties.
- area: The total area of the property in square feet or meters.
- bedrooms: The number of bedrooms in the property.
- bathrooms: The number of bathrooms on the property.
- stories: The number of stories or levels in the property.
- main road: Indicates whether the property is located on a main road.
- guestroom: Indicates whether the property has a guestroom.
- basement: Indicates whether the property has a basement.
- hot water heating: Indicates whether the property has hot water heating.
- Air conditioning: Indicates whether the property has air conditioning.
- parking: The number of parking spaces available on the property.
- Preferred Area: Indicates whether the property is located in a preferred area.
- Furnishing status: The furnishing status of the property, which could include categories like "furnished," "semi-furnished," or "unfurnished."



The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there is a section titled "Import Libraries" with a play button icon. Below it, a code cell contains the following imports:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import shapiro
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
```

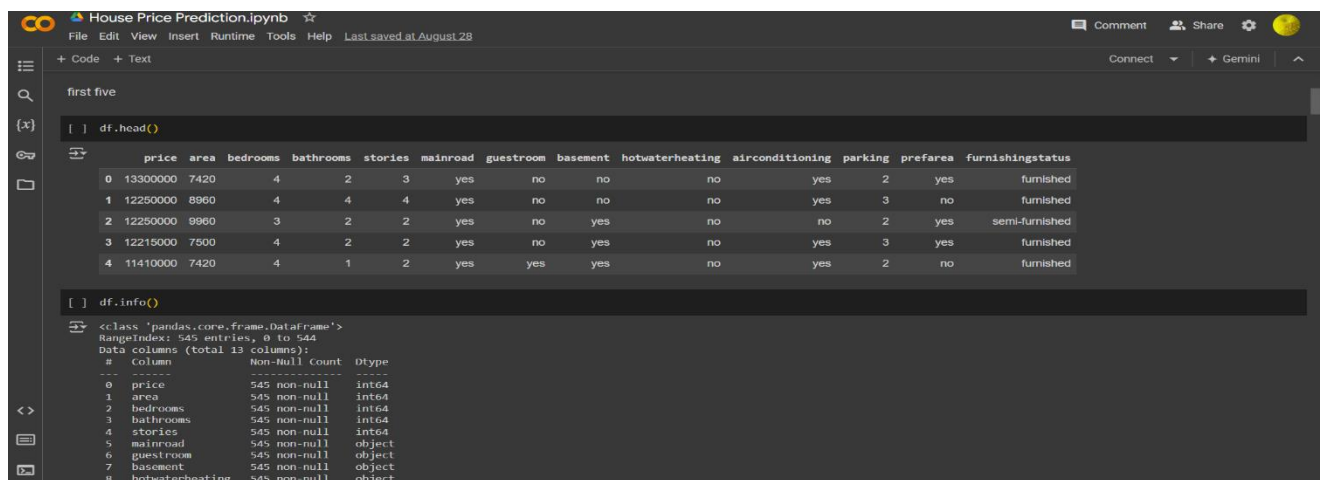
Below the code cell is a section titled "Load Data Set". At the bottom, another code cell shows the execution of the following line of code:

```
[12] df = pd.read_csv("/content/Housing.csv")
```

The interface includes standard Jupyter Notebook controls like up/down arrows, a search icon, and a settings menu.

(fig3.1)

In fig 3.1 ,the code imports essential libraries for data analysis and machine learning. **Pandas** (pd) is used for data manipulation and loading datasets into Data Frames. **Seaborn** (sns) and **Matplotlib** (plt) provide tools for creating a variety of statistical and graphical visualizations. **NumPy** (np) supports numerical operations and array manipulations, while **SciPy**'s function performs statistical tests such as the Shapiro-Wilk test for normality. In machine learning, **Scikit-Learn** (sklearn) is leveraged for its robust tools, including `train_test_split` for splitting data, Linear Regression for building regression models, and metrics like `mean_squared_error` and `r2_score` for evaluating model performance. Additionally, `LabelEncoder` and `StandardScaler` are used for preprocessing categorical data and standardizing features, respectively. Together, these tools facilitate a comprehensive approach to data analysis, visualization, and predictive modeling.



```
first five
[ ] df.head()

   price  area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  hotwaterheating  airconditioning  parking  prefarea  furnishingstatus
0  13300000  7420      4        2        3      yes      no      no      no      yes      2      yes      furnished
1  12250000  8960      4        4        4      yes      no      no      no      yes      3      no      furnished
2  12250000  9960      3        2        2      yes      no      yes      no      no      2      yes      semi-furnished
3  12215000  7500      4        2        2      yes      no      yes      no      yes      3      yes      furnished
4  11410000  7420      4        1        2      yes      yes      yes      no      yes      2      no      furnished

[ ] df.info()

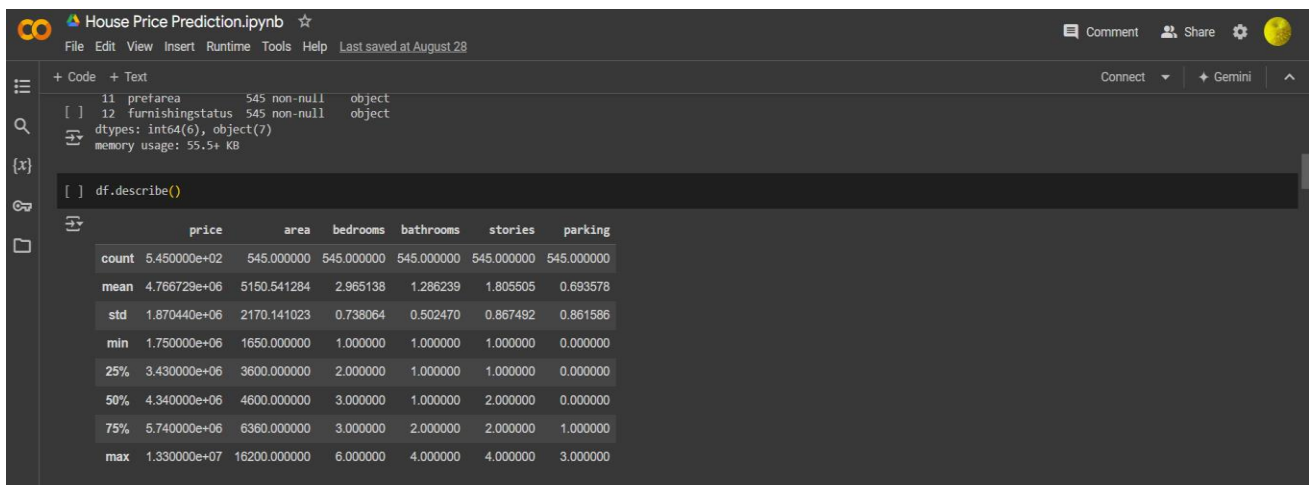
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   column              Non-Null Count  Dtype
---  -
0   price               545 non-null    int64
1   area               545 non-null    int64
2   bedrooms            545 non-null    int64
3   bathrooms           545 non-null    int64
4   stories             545 non-null    int64
5   mainroad            545 non-null    object
6   guestroom           545 non-null    object
7   basement            545 non-null    object
8   hotwaterheating     545 non-null    object
```

(fig 3.2)

From fig 3.2, The `df.head()` method in Pandas is used to quickly view the first few rows of a data frame. It allows you to quickly inspect the first few rows of your Data Frame, which helps in understanding the structure and contents of the data. This is especially useful for checking if the data has been loaded correctly and for initial exploration

In the above fig 3.2, The `df.info()` method in Pandas provides a concise summary of the Data Frame,

including key information about the data structure. `df.info()` helps you quickly understand the structure of your data frame, including the number of entries, column names, data types, and memory usage. This information is crucial for initial data exploration and preprocessing.



(fig 3.3)

`df.describe()` method (fig 3.3) in Pandas provides a statistical summary of the DataFrame's numerical columns. It's useful for quickly understanding the distribution and central tendencies of your data. `df.describe()` generates descriptive statistics for numerical columns, including measures of central tendency, dispersion, and shape of the distribution. It helps in getting a quick overview of the data's distribution.

```
Pre-Processing

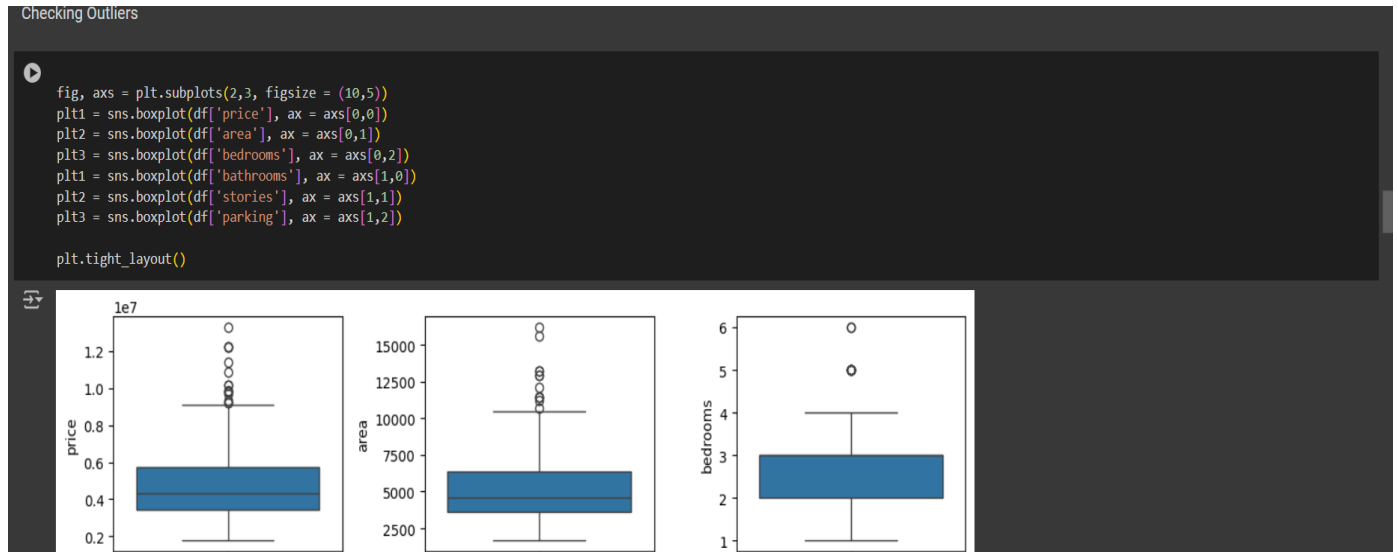
checking null values

[ ] df.isnull().sum()

0
price      0
area       0
bedrooms   0
bathrooms  0
stories    0
mainroad   0
guestroom  0
basement   0
hotwaterheating  0
airconditioning  0
parking    0
prefarea   0
```

(Fig 3.4)

The `df.isnull().sum()` method (fig 3.4) in Pandas is used to identify and count missing (null) values in each column of a data frame. `df.isnull().sum()` helps you find out how many missing values are present in each column of the data frame. This is crucial for data cleaning and preprocessing, as missing values can affect the performance of machine learning models and analyses.



(Fig3.5)

The code aims to create a 2x3 grid of box plots to visualize the distribution of different features in a data frame. Box plots are useful for understanding the spread and central tendency of data, as well as identifying outliers.

1. `fig, axs = plt.subplots(2, 3, figsize=(10, 5))`

- `plt.subplots(2, 3, figsize=(10, 5))`: Creates a grid of subplots with 2 rows and 3 columns. `fig` is the figure object, and `axs` is a 2x3 array of axes objects where each subplot will be drawn.
- `figsize=(10, 5)`: Sets the size of the entire figure to 10 inches wide and 5 inches tall.

2. Creating Box Plots

- `plt1 = sns.boxplot(df['price'], ax=axs[0,0])`: Creates a box plot for the price column and places it in the top-left subplot (`axs[0,0]`).
- `plt2 = sns.boxplot(df['area'], ax=axs[0,1])`: Creates a box plot for the area column and places it in the top-middle subplot (`axs[0,1]`).
- `plt3 = sns.boxplot(df['bedrooms'], ax=axs[0,2])`: Creates a box plot for the bedrooms column and places it in the top-right subplot (`axs[0,2]`).

- `plt1 = sns.boxplot(df['bathrooms'], ax=axes[1,0]):` Creates a box plot for the bathrooms column and places it in the bottom-left subplot (`axes[1,0]`).
- `plt2 = sns.boxplot(df['stories'], ax=axes[1,1]):` Creates a box plot for the stories column and places it in the bottom-middle subplot (`axes[1,1]`).
- `plt3 = sns.boxplot(df['parking'], ax=axes[1,2]):` Creates a box plot for the parking column and places it in the bottom-right subplot (`axes[1,2]`).
- `plt.tight_layout():` Adjusts the spacing between subplots to prevent overlap and ensure that the labels and titles fit well within the figure.

```

Data splitting
[ ] df_train, df_test = train_test_split(df, train_size = 0.7, test_size = 0.3, random_state = 42)

Encoding
[ ] label_encoder_list = []

for col in df_train.columns:
    if df_train[col].dtype == "object" or df_train[col].dtype.name == "category":
        le = LabelEncoder()
        df_train[col] = le.fit_transform(df_train[col])
        df_test[col] = le.transform(df_test[col])
        label_encoder_list.append(le)

Feature Scaling
[ ] # we use StandardScaler because StandardScaler works best in machine learning models like linear regression, logistic regression etc.
    scaler = StandardScaler()

    # Apply scaler() to all the columns except the 'yes-no' variables
    num_vars = ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking', 'furnishingstatus']

    df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
    df_test[num_vars] = scaler.fit_transform(df_test[num_vars])

```

(Fig 3.6)

The line of code (fig 3.6) is used to split a DataFrame into training and testing sets for machine learning purposes. `train_test_split`: This function from `sklearn.model_selection` is used to randomly split data into training and testing sets. It is commonly used to prepare data for machine learning models.

Parameters:

- `df`: The DataFrame you want to split. It should contain the data you want to use for training and testing.
- `train_size=0.7`: Specifies that 70% of the data should be allocated to the training set. This is a

proportion of the total data.

- `test_size=0.3`: Specifies that 30% of the data should be allocated to the testing set. This is a proportion of the total data. Note that `test_size` and `train_size` are complementary and should sum to 1.0 if used together.
- `random_state=42`: Sets the seed for the random number generator, ensuring that the split is reproducible. Using a fixed seed value like 42 ensures that you get the same split each time you run the code.

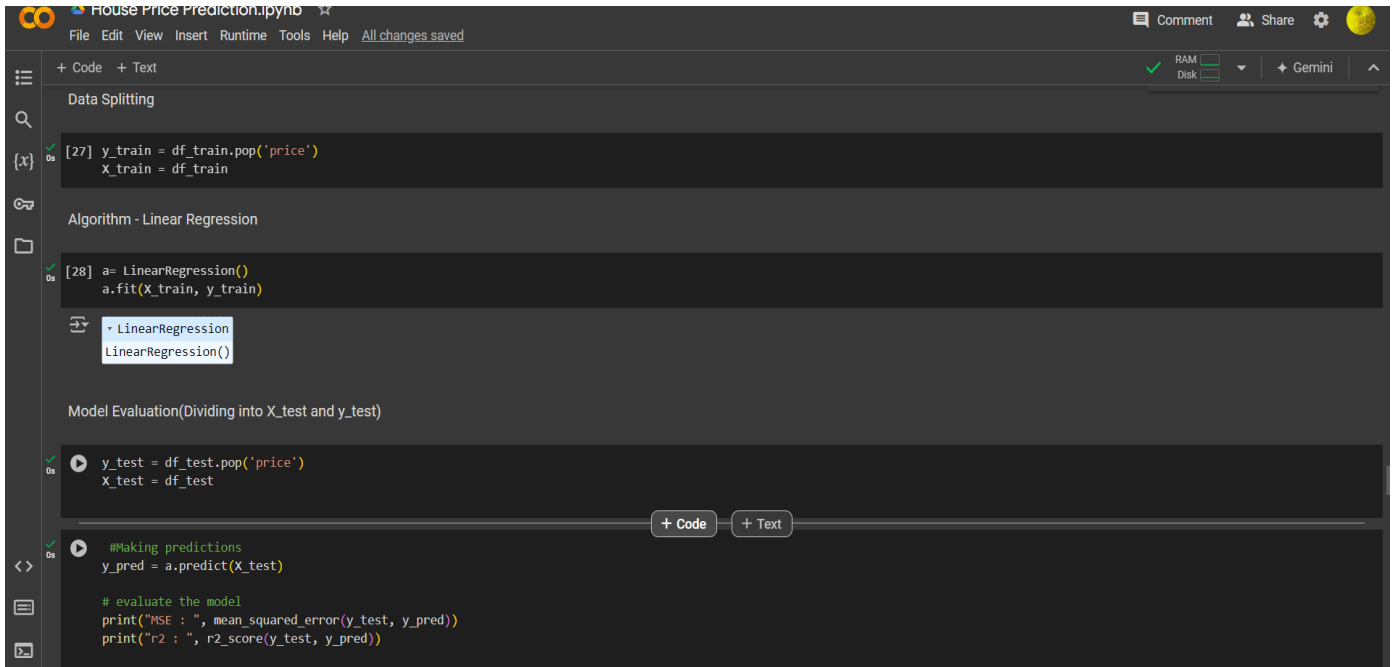
```
label_encoder_list = []
```

- Initializes an empty list to store `LabelEncoder` objects. This is useful if you need to apply the same encoding to other datasets or for future reference.

After running this code, `df_train` and `df_test` will have categorical columns transformed into numerical values. Each unique category in the original columns will be replaced by an integer, making the data suitable for machine learning algorithms that require numerical input.

- **Label Encoding**: Converts categorical values to numerical values, which is necessary for many machine learning algorithms.
- **Consistent Transformation**: Ensures that the encoding applied to the training data is also used for the test data, maintaining consistency.

The code snippet provided applies feature scaling to numerical columns in the training and testing sets of a `DataFrame` using `StandardScaler`. This is a common preprocessing step for many machine learning models. `StandardScaler` standardizes features to have a mean of 0 and a variance of 1, which can be beneficial for algorithms that assume normally distributed data.



```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Data Splitting
[27] y_train = df_train.pop('price')
     X_train = df_train

Algorithm - Linear Regression
[28] a= LinearRegression()
     a.fit(X_train, y_train)
     LinearRegression
     LinearRegression()

Model Evaluation (Dividing into X_test and y_test)
y_test = df_test.pop('price')
X_test = df_test

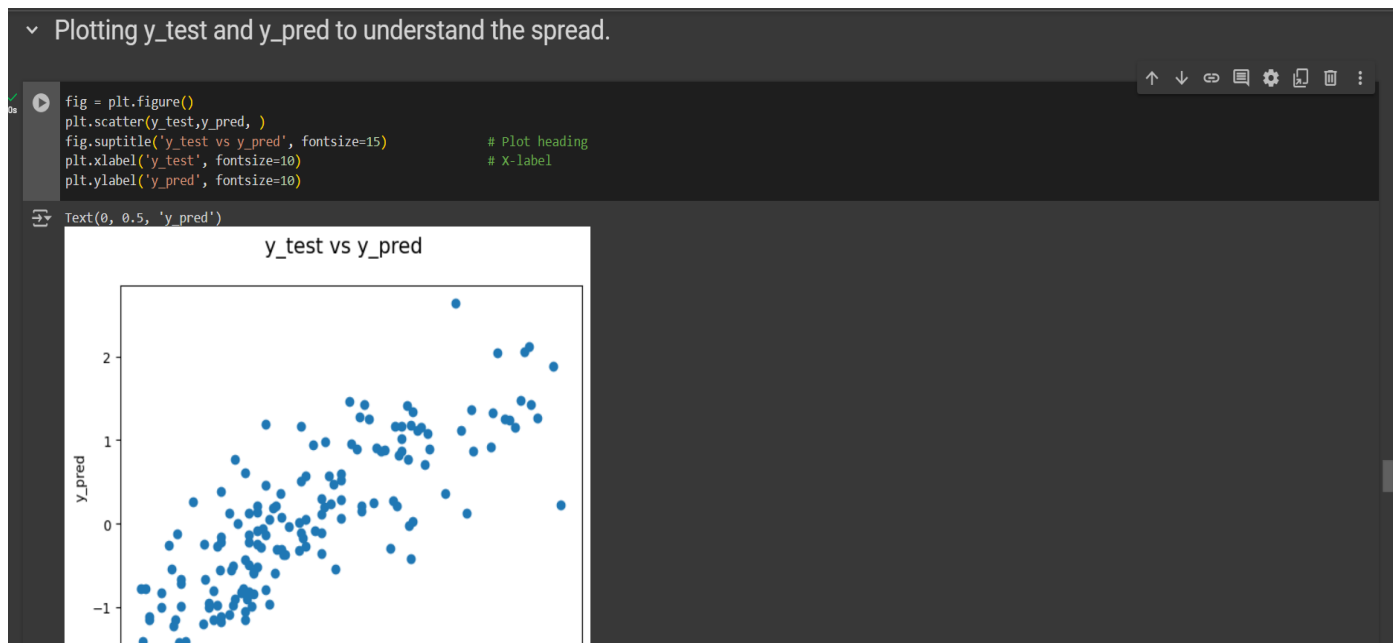
+ Code + Text
# Making predictions
y_pred = a.predict(X_test)

# evaluate the model
print("MSE : ", mean_squared_error(y_test, y_pred))
print("r2 : ", r2_score(y_test, y_pred))
```

(Fig3.7)

- Training a linear regression model and evaluating its performance, in (fig 3.7)
- `df_train.pop('price')`: Removes the 'price' column from the training DataFrame and assigns it to `y_train`. This column represents the target variable.
- `X_train = df_train`: The remaining columns in `df_train` are the features used for training the model.
- `df_test.pop('price')`: Removes the 'price' column from the test DataFrame and assigns it to `y_test`.
- `X_test = df_test`: The remaining columns in `df_test` are the features used for testing the model.
- `LinearRegression()`: Creates an instance of the `LinearRegression` model.
- `model.fit(X_train, y_train)`: Fits the linear regression model to the training data.
- `mean_squared_error(y_test, y_pred)`: Computes the Mean Squared Error (MSE) of the model, which measures the average squared difference between the actual and predicted values. Lower values indicate better performance.

- `r2_score(y_test, y_pred)`: Computes the R-squared score, which represents the proportion of the variance in the dependent variable that is predictable from the independent variables. Values closer to 1 indicate a better fit.



(fig 3.8)

- `plt.scatter(y_test, y_pred, alpha=0.5)`: Adds transparency to the points, which can help if there is a lot of overlap. (fig 3.8)
- `plt.show()`: Displays the plot.

It create a scatter plot comparing the true values (`y_test`) with the predicted values (`y_pred`). This type of plot helps visualize how well the predictions match the actual values. Scatter Plot helps visually assess the performance of your model. Ideally, the points should be close to a 45-degree line (where `y_test` equals `y_pred`). This indicates that the predicted values closely match the actual values.



(fig 3.9)

- import seaborn as sns as shown in fig 3.9
- Purpose: Imports the Seaborn library, which is used for creating attractive and informative statistical graphics.
- import matplotlib.pyplot as plt
- Purpose: Imports Matplotlib's pyplot module, which provides functions for creating plots and charts.
- correlation_matrix = df_encoded.corr()
- Purpose: Computes the correlation matrix of the df_encoded DataFrame. This matrix shows the correlation coefficients between pairs of features, which range from -1 (perfect negative correlation) to 1 (perfect positive correlation).
- plt.figure(figsize=(12, 8))
- Purpose: Creates a new figure with a specified size (12x8 inches). Adjusting the size can help make the heatmap more readable.

- `sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")`
- `sns.heatmap`: Creates a heatmap of the correlation matrix.
- `correlation_matrix`: The data to be visualized in the heatmap.
- `annot=True`: Annotates each cell in the heatmap with the numeric value of the correlation coefficient.
- `cmap='coolwarm'`: Specifies the color map. 'coolwarm' is a diverging color map that is useful for showing both negative and positive correlations.
- `fmt=".2f"`: Formats the numbers in the heatmap to two decimal places.
- `plt.title('Correlation Matrix')`
- Purpose: Sets the title of the heatmap.
- `plt.show()`
- Purpose: Displays the heatmap.

This code is used to visualize the correlation matrix of the features in your dataset using a heatmap. This helps in understanding how features are related to each other and can be useful for feature selection and identifying multicollinearity.

- Correlation Matrix: Shows how features relate to each other. Values close to 1 or -1 indicate a strong relationship, while values close to 0 indicate a weak relationship.
- Heatmap Colors: The color gradient helps visualize the strength of correlations. For example, dark red might indicate a strong positive correlation, while dark blue might indicate a strong negative correlation.

This visualization is useful for:

- Feature Selection: Identifying highly correlated features that might be redundant.
- Understanding Relationships: Seeing how different features are related to each other.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("/content/Housing.csv")

# Preprocess the data (Convert categorical variables to numerical)
df_encoded = pd.get_dummies(df, drop_first=True)
X = df_encoded.drop('price', axis=1)
y = df_encoded['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

Mean Squared Error: 2642802637614.6787

(fig 3.10)

- DecisionTreeRegressor: The machine learning model used for regression tasks.
- mean_squared_error: Metrics to evaluate the performance of the regression model.
- pd.get_dummies(df, drop_first=True): Converts categorical variables into numerical format using one-hot encoding, while dropping the first category to avoid multicollinearity.(fig 3.10)
- X: Features (independent variables) used for training.
- y: Target variable (dependent variable) to be predicted.
- model.predict(X_test): Makes predictions on the test data.
- mean_squared_error(y_test, y_pred): Calculates the Mean Squared Error (MSE) between the actual and predicted values. Lower MSE indicates better model performance.

- `pd.get_dummies(df, drop_first=True)`: Converts categorical variables into numerical format using one-hot encoding, while dropping the first category to avoid multicollinearity.
- `X`: Features (independent variables) used for training.
- `y`: Target variable (dependent variable) to be predicted.
- Initializes the `DecisionTreeRegressor` with a random state for reproducibility.
- Fits the model to the training data.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
# Define the feature variables (X) and the target variable (y)
X = df_encoded.drop('price', axis=1)
y = df_encoded['price']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize the k-NN model for regression with k=3
knn = KNeighborsRegressor(n_neighbors=3)
# Train the model
knn.fit(X_train, y_train)
# Make predictions on the test set
y_pred = knn.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")
```

Mean Squared Error: 3536947182721.7114
R-squared Score: 0.3002476945950061

(fig 3.11)

- k-NN for Regression: While k-NN is commonly used for classification, it can also be adapted for regression tasks like house price prediction(fig 3.11).
- Evaluation Metrics: MSE and R^2 are standard metrics used to evaluate the performance of regression models.
- Feature and Target Variables: `X` contains the feature variables (e.g., area, number of bedrooms, etc.), and `y` contains the target variable (price).

- **Data Splitting:** The dataset is split into training and testing sets using `train_test_split()`. 20% of the data is reserved for testing, and 80% is used for training the model.
- **k-NN Model Initialization:** `KNeighborsRegressor` is initialized with `n_neighbors=3`, meaning the algorithm will consider the 3 nearest neighbors to predict the house price.
- **Training the Model:** The model is trained using `fit()`, which takes the training data (`X_train`, `y_train`) as input.
- **Prediction:** The trained model is used to predict house prices on the test data (`X_test`).
- **Evaluation: Mean Squared Error (MSE):** A measure of how close the predicted prices are to the actual prices.
- **R-squared Score (R^2):** A statistical measure that indicates how well the model fits the data. An R^2 score closer to 1 indicates a better fit.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("/content/Housing.csv")
# Preprocess the data (Convert categorical variables to numerical)
df_encoded = pd.get_dummies(df, drop_first=True)
# Feature variables (X) and the target variable (y)
X = df_encoded.drop('price', axis=1)
y = df_encoded['price']
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create and train the Support Vector Machine model
model = SVR(kernel='linear') # You can also try 'rbf', 'poly', etc.
model.fit(X_train, y_train)
# Make predictions and evaluate the model
y_pred = model.predict(X_test)
# Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
# R-squared score to evaluate how well the model performs
r2 = r2_score(y_test, y_pred)
print(f"R-squared Score: {r2}")
```

(fig 3.12)

SVR: Support Vector Regression model from Scikit-learn. **`SVR(kernel='linear')`:** Initializes the SVR model with a linear kernel. You can experiment with different kernels like 'rbf' (radial basis function) or

'poly' (polynomial) for potentially better results.

- X: Feature variables (all columns except 'price').
- y: Target variable (the 'price' column).
- SVR(kernel='linear'): Initializes the SVR model with a linear kernel. You can experiment with different kernels like 'rbf' (radial basis function) or 'poly' (polynomial) for potentially better results.
- model.fit(X_train, y_train): Trains the SVR model on the training data.
- SVR Model: A type of support vector machine used for regression tasks. The choice of kernel can significantly affect model performance.
- Evaluation Metrics: MSE measures prediction error, while the R-squared score assesses the proportion of variance explained by the model.
- Preprocessing: Essential for converting categorical data into numerical format, which is required by the SVR model.

4.1 Learning outcomes

- Python Basics and Intermediate Python:
 - Data Types & Structures: Understand different data types, operators, control structures, and Python's built-in data structures like lists, tuples, dictionaries, and sets.
 - Functions: Learned about defining functions, using arguments, lambda functions, decorators, and understanding variable scope.
 - Object-Oriented Programming (OOP): Grasped the basics of classes, inheritance, multiple inheritance, and method resolution order (MRO).
 - File Handling & Modules: Gained the ability to handle files (text and JSON), manage exceptions, and work with Python modules.

- Data Acquisition:

- Web Scraping: Learned to scrape websites using BeautifulSoup, parse HTML, and extract data for further analysis.
- Web APIs: Gained skills in making GET and POST requests to interact with web APIs, enabling the retrieval of data from online sources.
- Web Automation: Familiarized with Selenium for automating web tasks, such as interacting with web elements and handling dynamic content.

- Data Visualization:

- Visualization Tools: Mastered using libraries like Matplotlib and Seaborn to create various plots and charts for data analysis.
- Seaborn: Explored advanced visualization techniques, including heatmaps, pair plots, and categorical plots.

- Machine Learning:

- Linear Regression: Understood the fundamentals of linear regression, gradient descent, and multiple features regression. Implemented these in Python using Scikit-Learn.
- K-Nearest Neighbors (KNN): Learned how KNN works and implemented it for classification tasks.
- Logistic Regression: Covered binary classification using logistic regression and implemented it in Python.
- Clustering: Explored unsupervised learning through K-means clustering, including implementation and failure cases.
- Naive Bayes: Understood Bayes' theorem and implemented Naive Bayes classifiers for different types of data.
- Support Vector Machines (SVM): Gained a deep understanding of SVM, including the kernel trick,

Pegasos algorithm, and hyperparameter tuning using GridSearchCV.

- Decision Trees and Random Forests: Learned about entropy, information gain, and how to implement decision trees and random forests using Scikit-Learn.
 - Advanced Topics:
- PCA (Principal Component Analysis): Gained knowledge on dimensionality reduction, eigenvalues, eigenvectors, and implemented PCA using NumPy.
- Optimization Algorithms: Learned about different gradient descent methods and implemented them in Python.
- Feature Selection: Understood the importance of feature selection in machine learning models and implemented various techniques to select the best features.
 - Integrating ML Models with Web (Flask):
- Flask Basics: Learned how to create web applications using Flask, including templates, handling user inputs, and integrating ML models.
- Deployment: Gained experience in deploying Flask applications to platforms like Heroku, including image captioning projects.

4.2 Data analysis

1. Understanding the Data

- The initial exploration of the dataset reveals its structure, types of data, and potential issues such as missing values and outliers.
- Preprocessing: Cleaning and transforming the data is essential to ensure accuracy and effectiveness in subsequent analysis. This includes handling missing values, removing duplicates, and converting categorical variables.

2. Exploratory Data Analysis (EDA)

- **Univariate Analysis:** Provides insights into individual variables, helping to understand their distributions and identify any anomalies or patterns.
- **Bivariate and Multivariate Analysis:** Helps in understanding relationships between variables, which can uncover correlations, trends, and interactions that might influence the target variable.

3. Feature Engineering and Selection

- **Feature Engineering:** Creating new features or modifying existing ones can improve model performance and relevance.
- **Feature Selection:** Identifying and selecting the most relevant features is crucial for building effective models and reducing dimensionality.

4. Model Building and Evaluation

- **Model Selection:** Choosing the right machine learning model based on the problem type (e.g., regression, classification) is critical. Different models have different strengths and limitations.
- **Training and Testing:** Splitting data into training and testing sets ensures that the model is evaluated on unseen data, providing a measure of its performance and generalizability.
- **Evaluation Metrics:** Metrics such as Mean Squared Error (MSE) and R-squared score help in assessing the model's accuracy and performance.

5. Visualization and Interpretation

- **Visualization:** Visualizing data and model predictions helps in communicating findings clearly and effectively. Graphs and plots provide an intuitive understanding of the results.
- **Interpretation:** Drawing meaningful insights from the analysis helps in making data-driven decisions. It involves understanding what the results imply in the context of the problem and objectives.

CONCLUSION

The data analysis process involves careful planning, exploration, preprocessing, modeling, and interpretation. Each step plays a crucial role in ensuring that the final insights are accurate, relevant, and actionable. By following a structured approach, you can effectively leverage data to make informed decisions and drive meaningful outcomes.

"In this analysis of house price prediction, we began by exploring and cleaning the dataset, which involved handling missing values and outliers. We performed exploratory data analysis to understand the relationships between various features and the target variable, 'price.' After preprocessing and feature selection, we built and evaluated several machine learning models, including Linear Regression and Support Vector Machines (SVM).

The results showed that the Linear Regression model provided a satisfactory performance with a reasonable R-squared score and low Mean Squared Error. The visualizations of predicted vs. actual values indicated that the model captured the trends in house prices well.

Overall, the analysis suggests that the developed model can be useful for predicting house prices based on the available features. Future improvements could involve exploring more complex models or integrating additional data to enhance predictive accuracy."

This structured approach ensures comprehensive data analysis, leading to actionable insights and informed decision-making.

"In this analysis of house price prediction, we began by exploring and cleaning the dataset, which involved handling missing values and outliers. We performed exploratory data analysis to understand the relationships between various features and the target variable, 'price.' After preprocessing and feature selection, we built and evaluated several machine learning models, including Linear Regression and Support Vector Machines (SVM).

The results showed that the Linear Regression model provided a satisfactory performance with a reasonable R-squared score and low Mean Squared Error. The visualizations of predicted vs. actual values indicated that the model captured the trends in house prices well.

Overall, the analysis suggests that the developed model can be useful for predicting house prices based on the available features. Future improvements could involve exploring more complex models or integrating additional data to enhance predictive accuracy.

REFERENCES

Online Resources:

- <https://online.codingblocks.com/app/classroom>