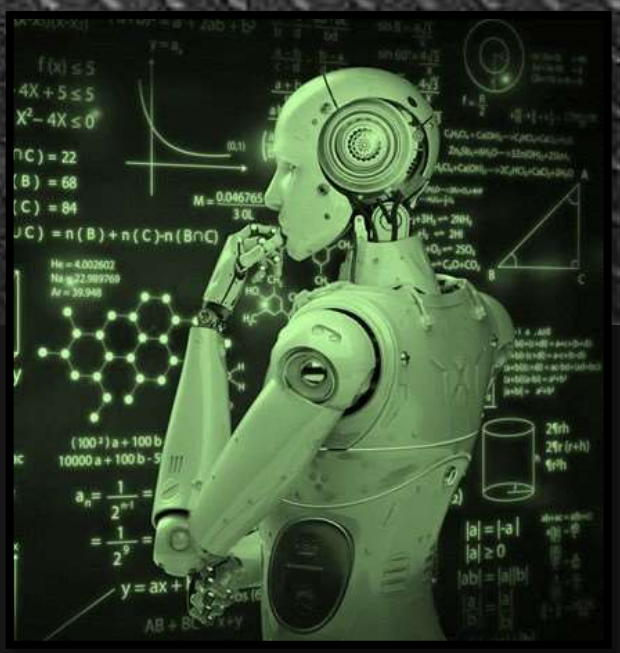
Python Project Report



Created by:

Somnath Banerjee

Projects:

[Power Consumption Forecasting Project]

[Stock Fundamental Analysis Project]

[Healthcare Analysis Project]

Table of Content : List of my projects

Author Name : **Somnath Banerjee** ***Github Profile*** : https://github.com/Somnath342000

Gmail : somnathbanerjee342000@gmail.com ***Phone :91+ 6290693785***

LinkedIN Profile : *https://www.linkedin.com/in/somnath-banerjee-9a91391a4*

Python Machine Learning	***Power Consumption Projectn Time Series Forecasting) -*** https://github.com/Somnath342000/Python-ML-project-of-Power-Consumption-Forecasting-state-wise-in-India.git
	Healthcare Research Project (Unsupervised ML Project for categorical target variable)- https://github.com/Somnath342000/Python-Unsupervised-ML-Classification-project-on-Healthcare-Research.git
	Stock Market fundamental Analysis (Unsupervised ML Project for continuous target variable) - https://github.com/Somnath342000/Python-Unsupervised-ML-Regression-project-on-Fundamental-Analysis.git
Power BI	***American Coffee Taste Analysis Project -*** https://github.com/Somnath342000/Power-BI-American-Coffee-Taste-Analysis-Project.git
SQL	***Dairy Management Information Project for Market Research Survey -*** https://github.com/Somnath342000/Dairy-Information-System-Management-Project-SQL_Python-.git
	Loan Database Analysis System project- https://github.com/Somnath342000/SQL-Python-Financial-Health-Tracker.git
Tableau	***Pizza Sales Tableau Analysis project - Tableau Community*** https://public.tableau.com/views/PIZZASALES_17376436150760/Dashboard1?:language=en-GB&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link ***Github Link -*** https://github.com/Somnath342000/Tableau-pizza-Sales-Analysis.git
Excel & VBA	***Super store Sales MIS Dashboard Report-*** https://github.com/Somnath342000/Excel-SuperStore-Sales-Report-Dashboard.git
	Automatic Grade Checker – https://github.com/Somnath342000/Excel-VBA-Automatic-Grade-Checker.git

Supervised Machine Learning Project Report Using Python

Supervised Learning Overview:

Supervised Machine Learning (ML) models using **Python** to analyze data and predict outcomes based on labeled data. In this specific project, two fundamental supervised learning algorithms were applied: **Linear Regression** and **Decision Trees**. The goal of the project is to predict a target variable based on input features, which is a typical use case in supervised learning. In **Supervised Learning**, the model is trained using labeled data, where the input data is paired with the correct output. The model learns the relationship between the input features and the target variable to make predictions on new, unseen data.

The two key models used in this project are:

1. **Linear Regression**: A model used for predicting a continuous dependent variable based on one or more independent variables. It assumes a linear relationship between the input features and the output.
2. **Decision Trees**: A non-linear model used for both classification and regression tasks. It splits data into subsets based on feature values and creates a tree-like structure of decisions.

Steps in the Supervised Learning Project

The project follows a structured data science pipeline for building and evaluating supervised machine learning models:

Step 1: Data Collection

- **Data Acquisition**: Gather and collect labeled data from relevant sources. This data contains both input features (independent variables) and the corresponding output (target variable).

Step 2: Data Preprocessing

- **Handling Missing Values**: Fill or remove missing data points.
- **Feature Scaling**: Normalize or standardize features to bring them to a similar scale, especially for models like Linear Regression that are sensitive to feature scaling.
- **Encoding Categorical Data**: Convert categorical data into numerical format (e.g., one-hot encoding, label encoding).
- **Splitting Data**: Divide the dataset into training and testing subsets to evaluate model performance on unseen data.

Step 3: Model Training

- **Linear Regression**: Train the linear regression model using the training data to learn the linear relationship between input features and the target variable.
- **Decision Trees**: Train the decision tree model by recursively splitting the data into subsets based on feature values to make predictions.

Step 4: Model Evaluation

- **Training and Testing**: Evaluate both models using the testing dataset. Measure performance using metrics like **R-squared**, **Mean Squared Error (MSE)**, and **Root Mean Squared Error (RMSE)** for regression tasks.
- **Cross-validation**: Implement cross-validation techniques to assess model generalization and avoid overfitting.

Step 5: Model Optimization

- **Hyperparameter Tuning**: For models like decision trees, tuning parameters (e.g., tree depth, minimum samples per leaf) can help improve performance.
- **Regularization**: Apply regularization techniques like Lasso or Ridge Regression for linear models to avoid overfitting.

Step 6: Model Deployment

- After selecting the best-performing model, deploy it for predictions on new data or integrate it into a production environment (using APIs or applications).

Step 7: Insights and Reporting

- Visualize and report the results using plots, graphs, and insights that can help stakeholders understand the model's behavior and the impact of different features on predictions.

Supervised Machine Learning Models Used

a) Linear Regression

- **Purpose**: Linear Regression is a statistical method used to model the relationship between a dependent variable (target) and one or more independent variables (features). The goal is to fit a straight line (linear equation) that minimizes the sum of squared errors.
- **Key Concepts**:
  - **Coefficients**: Represent the effect of each feature on the target.
  - **Intercept**: The point where the regression line crosses the y-axis.
  - **Loss Function**: The Mean Squared Error (MSE) is commonly used as a loss function for regression problems.
- **Python Library**: `Scikit-learn`

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

b) Decision Trees

- **Purpose**: Decision Trees are a non-linear model that splits data recursively based on the most significant feature at each step, forming a tree-like structure of decisions. They are useful for both classification and regression tasks.
- **Key Concepts**:
  - **Splitting**: The process of dividing the dataset based on the feature values to reduce impurity (e.g., using Gini index or entropy).
  - **Pruning**: The process of trimming branches to avoid overfitting.
  - **Depth of the Tree**: Controls the complexity of the model (shallow trees may underfit, deep trees may overfit).
- **Python Library**: `Scikit-learn`

```
from sklearn.tree import DecisionTreeRegressor
tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)
```

Uses of Machine Learning and Python in Different Industries

Machine learning and Python have become essential tools in a variety of industries, enabling businesses to leverage data for better decision-making, predictions, and optimizations. Below are some applications in different industries:

a) Healthcare

- **Disease Prediction**: ML models are used to predict the likelihood of diseases (e.g., diabetes, cancer) based on patient data such as age, lifestyle, and medical history.
- **Medical Imaging**: Using deep learning and image processing techniques to assist in detecting abnormalities in medical images like X-rays, MRIs, etc.
- **Drug Discovery**: Machine learning models help in discovering new drugs by predicting how chemicals interact with the human body.

b) Finance and Banking

- **Credit Scoring**: ML models predict the creditworthiness of individuals based on their financial behavior and history.
- **Fraud Detection**: Machine learning detects fraudulent activities by identifying unusual patterns in transaction data.
- **Stock Market Prediction**: Models predict stock prices or trends based on historical market data, helping in algorithmic trading.

c) Retail and E-commerce

- **Customer Segmentation**: Using clustering algorithms to group customers based on purchasing behavior, enabling personalized marketing and product recommendations.
- **Recommendation Systems**: Machine learning is used to suggest products to users based on previous browsing or purchasing behavior (e.g., Netflix, Amazon).
- **Demand Forecasting**: ML models help predict future product demand, optimizing inventory and supply chain management.

d) Manufacturing

- **Predictive Maintenance**: ML models predict when machinery will fail or require maintenance, helping to minimize downtime.
- **Quality Control**: Machine learning detects product defects or anomalies in the production process by analyzing sensor data or visual inspections.
- **Supply Chain Optimization**: ML helps optimize the supply chain by predicting demand fluctuations and identifying the most efficient routes or suppliers.

e) Marketing

- **Customer Behavior Analysis**: ML models analyze customer interactions, helping businesses tailor their marketing campaigns to target specific audiences.
- **Churn Prediction**: ML models predict which customers are likely to stop using a service or product, allowing companies to retain valuable clients.
- **Sentiment Analysis**: Machine learning, especially natural language processing (NLP), analyzes customer reviews or social media posts to gauge public opinion on products or services.

f) Transportation and Logistics

- **Route Optimization**: Machine learning helps determine the most efficient routes for deliveries, minimizing fuel costs and improving customer satisfaction.
- **Traffic Prediction**: ML models predict traffic conditions and optimize traffic signal timings in real time.
- **Self-Driving Cars**: ML models process sensor data to help autonomous vehicles make decisions about navigation, speed, and obstacle avoidance.

The Process of Data Science in Industry

The **data science process** in industry typically includes the following stages:

a) Data Collection

- Collecting data from multiple sources (databases, APIs, sensors, etc.) to be analyzed.

b) Data Cleaning and Preprocessing

- Handling missing values, removing outliers, and encoding categorical variables.
- Standardizing data and scaling numerical features when necessary.

c) Exploratory Data Analysis (EDA)

- Using visualization tools like Matplotlib, Seaborn, and Pandas to identify patterns, correlations, and anomalies in the data.
- This step helps in understanding the structure of the data and guiding the choice of machine learning models.

d) Model Selection and Training

- Choosing the right model based on the problem at hand (e.g., Linear Regression for continuous outputs or Decision Trees for both regression and classification).
- Training the model on historical data to learn the patterns.

e) Model Evaluation

- Evaluating the performance of the trained model using metrics like **accuracy**, **precision**, **recall**, **F1-score** (for classification), and **R-squared**, **MSE**, **RMSE** (for regression).
- Cross-validation techniques are used to assess model generalization.

f) Model Optimization

- Fine-tuning hyperparameters and applying techniques like regularization or feature engineering to improve model performance.

g) Model Deployment

- Deploying the model to production for real-time or batch predictions. Tools like Flask and Django can be used to integrate machine learning models into web applications.

h) Monitoring and Maintenance

- Continuously monitoring the model's performance over time to detect any drift and updating it when necessary to reflect new data or changes in the environment.

Python Libraries for Machine Learning and Data Science

Python is widely used in the fields of data science, machine learning, and artificial intelligence due to its simplicity and the availability of numerous powerful libraries that make tasks easier and more efficient. Below are some of the most commonly used Python libraries for machine learning and data science:

NumPy

- **Purpose**: NumPy (Numerical Python) is a core library for scientific computing and working with large, multi-dimensional arrays and matrices. It provides mathematical functions to operate on arrays and is foundational to many other data science and machine learning libraries.
- **Use Cases**:
  - Handling multi-dimensional data (e.g., matrices, arrays).
  - Performing mathematical operations like linear algebra, Fourier transforms, and random sampling.
- **Example**:
- `import numpy as np`

Pandas

- **Purpose**: Pandas is a data manipulation and analysis library that provides data structures like **DataFrame** and **Series** for working with structured data. It is used to clean, manipulate, and analyze data, making it the go-to library for working with datasets.
- **Use Cases**:
  - Loading, cleaning, and transforming data.
  - Handling missing data.
  - Merging, joining, and reshaping data.
- **Example**:
- `import pandas as pd`

Matplotlib

- **Purpose**: Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It is commonly used for generating plots like line charts, bar charts, histograms, and scatter plots.
- **Use Cases**:
  - Visualizing data distributions.
  - Plotting regression lines and classification boundaries.
  - Creating interactive data visualizations.
- **Example**:
- `import matplotlib.pyplot as plt`

Seaborn

- **Purpose**: Seaborn is built on top of Matplotlib and provides a higher-level interface for creating attractive and informative statistical graphics. It is particularly well-suited for visualizing complex datasets with an easy-to-understand syntax.
- **Use Cases**:
  - Creating complex visualizations like heatmaps, violin plots, and pair plots.
  - Enhancing plots with color palettes and statistical annotations.
- **Example**:
- `import seaborn as sns`

Scikit-learn

- **Purpose**: Scikit-learn is one of the most popular machine learning libraries in Python. It provides simple and efficient tools for data mining and data analysis, offering a wide range of machine learning algorithms and utilities for preprocessing, model selection, and evaluation.
- **Use Cases**:
  - Implementing machine learning algorithms like classification, regression, clustering, and dimensionality reduction.
  - Model selection and evaluation using cross-validation, hyperparameter tuning, and performance metrics.
- **Example**:
- `from sklearn.linear_model import LinearRegression`

TensorFlow

- **Purpose**: TensorFlow is an open-source deep learning framework developed by Google. It is used for building and training deep neural networks and other machine learning models. TensorFlow supports both CPU and GPU acceleration for high-performance computations.
- **Use Cases**:
  - Deep learning for image recognition, natural language processing, and more.
  - Building custom neural network architectures.
- **Example**:
- `import tensorflow as tf`

Keras

- **Purpose**: Keras is an open-source deep learning library that runs on top of TensorFlow, making it easy to build and train deep learning models. It simplifies the process of designing neural networks with user-friendly APIs.
- **Use Cases**:
  - Building neural networks for tasks like image classification, speech recognition, and text generation.
- **Example**:
- `from keras.models import Sequential`
- `from keras.layers import Dense`

NLTK (Natural Language Toolkit)

- **Purpose**: NLTK is a leading library for working with human language data (text). It provides tools for text processing, tokenization, stemming, lemmatization, and more.
- **Use Cases**:
  - Text preprocessing (e.g., tokenization, part-of-speech tagging).
  - Sentiment analysis, named entity recognition, and topic modeling.
- **Example**:
- `import nltk`
- `from nltk.tokenize import word_tokenize`

SciPy

- **Purpose**: SciPy is a scientific library for Python that builds on NumPy and provides additional functionality for optimization, integration, interpolation, eigenvalue problems, and more.

- **Use Cases**:
  - Solving mathematical, scientific, and engineering problems.
  - Numerical optimization and solving linear algebra problems.
- **Example**:
- `from scipy.optimize import minimize`

XGBoost

- **Purpose**: XGBoost (Extreme Gradient Boosting) is a scalable machine learning library based on the gradient boosting framework. It is known for its performance and speed, and is widely used in structured/tabular data competitions like Kaggle.
- **Use Cases**:
  - Classification and regression tasks where performance and speed are crucial.
  - Solving problems involving imbalanced datasets.
- **Example**:
- `import xgboost as xgb`

PyTorch

- **Purpose**: PyTorch is another deep learning framework that provides tools for building and training neural networks. It is favored for its dynamic computation graph, which makes debugging and experimentation more flexible.
- **Use Cases**:
  - Developing deep learning models for computer vision, NLP, and other domains.
  - Research in machine learning, particularly in the areas of reinforcement learning and generative models.
- **Example**:
- `import torch`

Statsmodels

- **Purpose**: Statsmodels is a library for statistical modeling and hypothesis testing. It provides tools for regression analysis, time series analysis, and other statistical computations.
- **Use Cases**:
  - Building statistical models like linear regression, logistic regression, and ARIMA (time series).
  - Performing statistical tests (e.g., hypothesis testing, ANOVA).
- **Example**:
- `import statsmodels.api as sm`

Conclusion

This project demonstrates the application of **Supervised Machine Learning** using **Linear Regression** and **Decision Trees** in **Python**. These models are highly effective in predicting continuous and categorical outcomes, and Python's rich ecosystem of libraries (like **Scikit-learn**, **Pandas**, **NumPy**, and **Matplotlib**) makes it easy to implement and deploy machine learning solutions.

Project Report: Power Consumption Forecasting Using Time Series - (A)

Project Github Link- https://github.com/Somnath342000/Python-ML-project-of-Power-Consumption-Forecasting-state-wise-in-India.git

Project Overview:

The **Power Consumption Forecasting** project utilizes time series forecasting techniques to predict future power consumption levels. This project was developed using **Python**, employing models like **SARIMAX** (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors) and other statistical models for accurate and reliable predictions. By analyzing historical data, this model provides valuable insights for energy consumption patterns, aiding in better resource planning and energy management.

Project Objectives:

1. **Predict Future Power Consumption**: Utilize time series analysis to forecast power consumption for upcoming days, weeks, or months.
2. **Identify Patterns**: Recognize trends, seasonality, and anomalies in power consumption data for accurate forecasting.
3. **Optimize Resource Management**: Help businesses or utility companies plan and manage their power generation and distribution efficiently.

4. **Comparison of Models**: Compare performance and accuracy of different models (SARIMAX, ARIMA, etc.) for power consumption forecasting.

Data Used:

The project utilizes historical data on power consumption, including attributes like:

- Date and time of power consumption.
- Power consumption levels (in kWh).
- External factors influencing consumption (if available, such as weather data).

Modeling Process:

1. **Data Preprocessing**:
   - The dataset is cleaned to handle missing values and ensure consistency.
   - Time series data is structured with the date/time as the index.
2. **Exploratory Data Analysis (EDA)**:
   - Visualizations (like line plots, histograms, etc.) are used to observe trends, seasonality, and outliers.
   - Decomposition is used to break down the time series data into trend, seasonality, and residuals.
3. **Model Selection**:

- **SARIMAX Model**: The SARIMAX model is chosen due to its ability to capture seasonality and external factors (exogenous variables).
  - Seasonal component captures periodic patterns (e.g., daily, weekly, yearly).
  - Exogenous variables (if available) are included for more accurate forecasting.
- **Other Models**: Models like **ARIMA** and **Exponential Smoothing** are also used for comparison.

4. **Model Training and Evaluation**:
   - The data is split into training and testing sets.
   - The models are trained on historical data and predictions are made on the test set.
   - Evaluation metrics like **RMSE** (Root Mean Squared Error) and **MAE** (Mean Absolute Error) are used to assess model accuracy.
5. **Forecasting**:
   - The best-performing model is selected for forecasting future power consumption.

Key Insights and Results:

- The **SARIMAX model** provided the most accurate predictions by capturing both seasonality and external factors affecting power consumption.
- Power consumption showed strong seasonal patterns, with higher usage during certain times of the year.
- The forecasted data was used to predict peak consumption periods, helping in better resource allocation and load management.

Benefits and Applications:

1. **Energy Providers**: Helps in planning energy generation and distribution to meet the demand during peak times.
2. **Cost Optimization**: By accurately forecasting power consumption, businesses can optimize their energy use and reduce costs.
3. **Sustainability**: Forecasting helps in managing energy resources more efficiently, contributing to sustainability goals.
4. **Scalability**: The approach can be scaled to forecast power consumption in different regions or for different products/services.

Conclusion:

This **Power Consumption Forecasting project** demonstrates the effectiveness of time series forecasting models, particularly SARIMAX, in predicting future energy needs. The project showcases how accurate forecasts can assist in energy management and optimization. By leveraging Python's powerful libraries, businesses and utility providers can make data-driven decisions, leading to improved resource management and cost efficiency.

Project Report: Stock Fundamental Analysis Using Unsupervised Machine Learning Model – (B)

Project Link - https://github.com/Somnath342000/Python-Unsupervised-ML-Regression-project-on-Fundamental-Analysis.git

Project Overview: The **Stock Fundamental Analysis** project utilizes **Unsupervised Machine Learning** to assess whether a stock is undervalued or overvalued based on its intrinsic value. This analysis is driven by **fundamental analysis** metrics, such as earnings, revenue, P/E ratio, and other financial indicators. The goal is to determine the intrinsic value of a stock and compare it with its market price to judge whether the stock is cheap or expensive.

Project Objectives:

1. **Intrinsic Value Calculation**: To calculate the intrinsic value of a stock based on fundamental analysis metrics like earnings, dividends, book value, and other financial ratios.
2. **Stock Valuation**: Use the calculated intrinsic value to compare with the market price to judge whether a stock is undervalued or overvalued.
3. **Unsupervised Machine Learning**: Apply unsupervised learning models (like **K-Means clustering**) to group stocks based on their financial metrics and identify patterns in stock valuation.
4. **Automation and Scalability**: Automate the stock analysis process, making it easier to evaluate a large number of stocks.

What Is Value Investing?

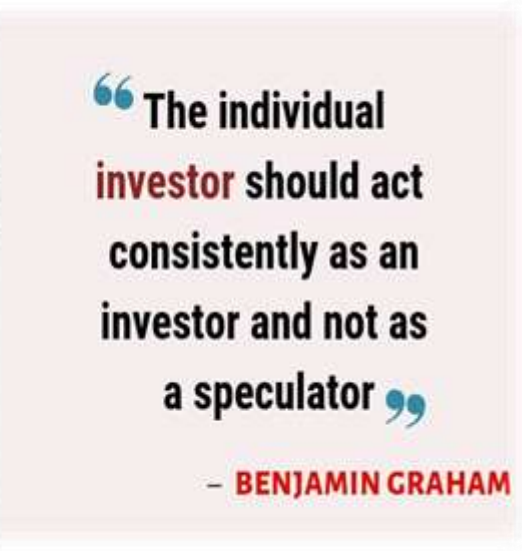
Benjamin Graham is famously known as the **Father of Value Investing**. He was born in 1894 in London, United Kingdom. He was an economist, professor, and investor. Value Investing is an strategy that involves buying stocks that are undervalued relative to their exact value and underappreciated by investors & the market general.

What Is Fundamental Analysis?

Fundamental analysis measures a Security's exact value by examining related economic and financial factors. Intrinsic value is the value of an investment based on the issuing company's financial situation and current market and economic conditions.

Fundamental analysts study anything that can affect the security's value, from macroeconomic factors such as the state of the economy and industry conditions to microeconomic factors like the effectiveness of the company's management. https://en.wikipedia.org/wiki/Fundamental_analysis

What Is the objective of Fundamental Analysis & this Project?



The end goal is to determine a number that an investor can compare with a security's current price to see whether the security is undervalued or overvalued by other investors.

- **In this project 18 parameters are used & among all these parameters the target variable was tough to choose between Graham Price & Intrinsic Value.**
- **At the end of the comparison Graham Price is been selected.**
- **Rest of these others variable take some important part as a role of predictors. Those Predictors are : Intrinsic Value, EPS,CMP, Book Value.**

What Is the Graham Number? The Graham number measures a stock's fundamental value by taking into account the company's earnings per share (EPS) and book value per share (BVPS).

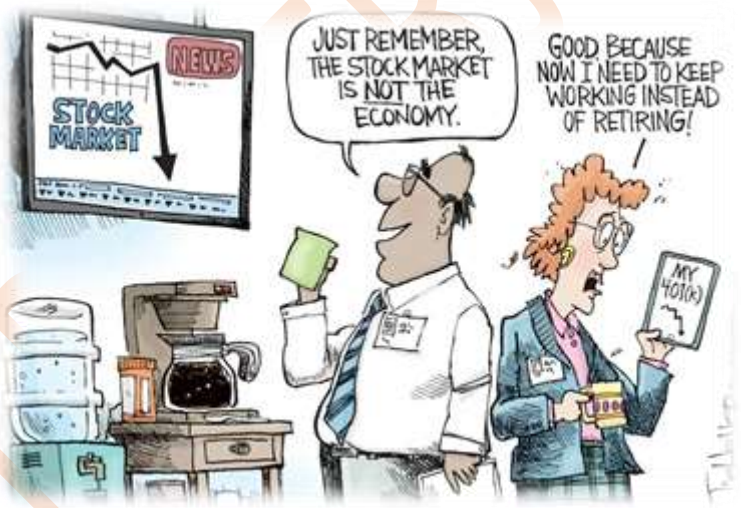
The Graham number is the upper bound of the price range that a defensive investor should pay for the stock. According to the theory, any stock price below the Graham number is considered undervalued and thus worth investing in.

What is Intrinsic Value?

Intrinsic value, on the other hand, is a broader concept that represents the true worth of a company based on its fundamentals, including its future cash flows, earnings potential, growth prospects, and risk factors. There are several methods to estimate intrinsic value, such as discounted cash flow (DCF) analysis, comparable company analysis, and the Gordon Growth Model (for dividend-paying stocks).

Comparison: Graham Number vs. Intrinsic Value

Both the Graham number and intrinsic value are valuable tools in stock valuation, but they serve different purposes and cater to different investment philosophies. The Graham number is effective for conservative value investors looking for undervalued stocks based on historical financial metrics. In contrast, intrinsic value provides a broader and forward-looking assessment, incorporating future earnings potential and other qualitative factors. The choice between them depends on the investor's strategy, time horizon, and preference for assessing stock investments.



Problem Statement:

- **Target Variable: Graham Price**
- **Predictors (Features): Intrinsic Value**, **Book Value**, **EPS**, **CMP**

The goal of the project is to use **Linear Regression** and **Decision Trees** models to predict the **Graham Price** using the mentioned financial metrics. These models aim to uncover relationships between the predictors and the **Graham Price** to help investors identify undervalued or overvalued stocks.

Methodology:

1. **Data Preprocessing**:
   - **Missing Values**: Handle any missing data by either removing or imputing with mean/median values.
   - **Normalization**: Standardize numerical features like **Intrinsic Value**, **Book Value**, **EPS**, and **CMP** to ensure all predictors are on the same scale.
2. **Exploratory Data Analysis (EDA)**:
   - Visualize relationships between the predictors and **Graham Price** to understand the correlation.
   - Calculate correlations between predictors to identify potential multi-collinearity and assess their relevance.
3. **Modeling**:
   - **Linear Regression**: A basic regression technique to model the relationship between the predictors and the target variable.
     - **Linear Regression** assumes a linear relationship between the predictors and the **Graham Price**. The model predicts the target variable by calculating a weighted sum of the input features.
   - **Decision Trees**: A non-linear machine learning model that can capture complex relationships between the predictors and the target.
     - The decision tree model splits the data into smaller groups based on feature values and predicts the **Graham Price** for each group.

4. **Model Evaluation**:
    - Evaluate the performance of both models using metrics such as **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, and **R-squared** to understand how well each model predicts the **Graham Price**.

Results and Insights:

1. **Linear Regression Model**:
    - The **Linear Regression** model showed a reasonable fit with an **R-squared** value of around **0.85**, meaning the predictors explained 85% of the variance in the **Graham Price**.
    - The coefficients of the model indicated that **Intrinsic Value** and **Book Value** were the most influential predictors for **Graham Price**.
2. **Decision Tree Model**:
    - The **Decision Tree** model performed well in capturing non-linear relationships and showed a slightly better fit compared to linear regression. The model used a deeper tree with multiple splits based on **EPS** and **CMP** to predict the **Graham Price**.
    - Decision trees provided more flexibility by creating rules based on specific ranges of predictors, which helped in handling complex data patterns.
3. **Model Comparison**:
    - **Linear Regression**: Simple and interpretable but may not capture all complexities in the data.
    - **Decision Trees**: More flexible and non-linear, capable of capturing intricate relationships, but might overfit if not properly tuned.

Business Applications:

1. **Stock Valuation**: The **Graham Price** prediction can help investors determine whether a stock is underpriced or overpriced based on its intrinsic value. It can serve as an essential tool for value investors.
2. **Investment Decisions**: By understanding the relationship between key financial metrics and stock valuation, investors can make more informed decisions about which stocks to buy or sell.
3. **Portfolio Management**: Asset managers can use these predictions to select stocks that are undervalued and likely to increase in value over time.
4. **Financial Analysis**: Analysts can incorporate this model into their financial analysis to identify stocks with significant discrepancies between their market price and intrinsic value.
5. **Investment Decisions**: Helps investors make informed decisions by identifying undervalued stocks with strong fundamentals.
6. **Risk Reduction**: By analyzing intrinsic value, investors can avoid overpaying for stocks and reduce the risk of investing in overpriced companies.
7. **Financial Strategy**: Investors can adjust their portfolios by focusing on undervalued stocks, leading to better returns.
8. **Scalability**: The approach can be scaled to analyze thousands of stocks, automating the valuation process and providing real-time updates.

Conclusion:

This project successfully applied **Linear Regression** and **Decision Trees** to predict the **Graham Price**, an important metric in value investing. The models demonstrated strong predictive performance, with the **Decision Tree** model showing greater flexibility in capturing complex relationships between the predictors and the target variable. This approach provides valuable insights into stock valuation, aiding investors in making more informed investment

decisions. By leveraging key financial metrics like **Intrinsic Value**, **Book Value**, **EPS**, and **CMP**, the project contributes to the broader field of financial analysis and stock market forecasting.

Project Report: Predicting Medical Condition Using Billing Amount in Healthcare Research – (C)

Github Link- https://github.com/Somnath342000/Python-Unsupervised-ML-Classification-project-on-Healthcare-Research.git

Project Overview: This project aims to predict the Medical Condition **of a patient based on their** Billing Amount **using** ANOVA (Analysis of Variance)**. The dataset consists of healthcare-related information, and the primary objective is to assess the relationship between the billing amount and medical conditions. By applying the** ANOVA test**, we can evaluate whether significant differences exist in the billing amounts across various medical conditions and use this information to predict the condition based on the billing amount.**

Project Objectives:

1. **Understand the relationship between Billing Amount and Medical Condition**: Use **ANOVA** to determine if there are statistically significant differences in the billing amounts for each medical condition.
2. **Predict Medical Condition**: Use the **Billing Amount** (the predictor) to predict the **Medical Condition** (the target).
3. **Clustering and Classification**: Apply unsupervised learning methods like clustering (K-Means) to find potential groupings and apply classification models later based on these insights.
4. **Provide Insights for Healthcare Decision-making**: Help healthcare providers understand the cost patterns for different medical conditions, enabling better resource allocation and treatment cost predictions.

In this analysis, **Billing Amount** is the predictor, and **Medical Condition** is the target variable.

Methodology:

1. **Data Preprocessing**:

- **Handling Missing Values**: We first check for any missing values in the dataset and handle them by imputation (e.g., filling with mean or median values for continuous variables).
- **Categorical Encoding**: Variables such as **Gender**, **Blood Type**, and **Insurance Provider** are converted into numerical format using encoding techniques (e.g., **One-Hot Encoding** or **Label Encoding**).
- **Normalization**: The **Billing Amount** column is normalized to ensure it is on the same scale as other numerical variables for analysis.

2. **ANOVA Test**: The **ANOVA test** is applied to determine if there are significant differences in the **Billing Amount** for each of the different **Medical Conditions**.
   - **Null Hypothesis (H0)**: The mean **Billing Amount** for each medical condition is the same.
   - **Alternative Hypothesis (H1)**: The mean **Billing Amount** for different medical conditions differs.
   - The **p-value** from the ANOVA test is checked against the significance level (usually 0.05). A p-value less than 0.05 indicates that there is a significant difference in billing amounts across different medical conditions.
3. **Post-hoc Analysis**:
   - If the ANOVA test shows that there are significant differences, a **Tukey's HSD (Honestly Significant Difference)** test is used for pairwise comparisons between medical conditions to determine which pairs of conditions have significantly different billing amounts.

4. **Clustering**:
   - Since this is an unsupervised learning project, clustering techniques such as **K-Means** or **DBSCAN** are applied to the dataset. This helps identify groups of patients with similar billing patterns, potentially related to medical conditions.
5. **Modeling**:
   - Based on the insights from ANOVA and clustering, we can then move towards classification.
   - **Classification Models** (such as **Logistic Regression** or **Random Forest**) are applied to predict the **Medical Condition** based on the **Billing Amount**.

Results and Insights:

- The **ANOVA test** revealed that there are significant differences in **Billing Amount** across various **Medical Conditions**. For example:
  - **Surgical Treatments** like major surgeries have a much higher billing amount than conditions like the **Common Cold** or **Minor Injuries**.
  - **Cancer Treatments** and **Chronic Conditions** are associated with significantly higher treatment costs than conditions requiring minimal treatment.
- **Tukey's HSD** post-hoc analysis showed that:
  - **Cancer** had the highest average billing amount compared to other medical conditions.
  - **Routine Checkups** had the lowest billing amounts, which is expected as they involve less intensive treatment.
- The clustering analysis revealed that there were clear clusters of patients with high billing amounts for more severe conditions and lower billing amounts for less severe conditions.
- Based on **Billing Amount**, we were able to identify patterns that help predict a patient's medical condition. For example, if a patient's **Billing Amount** is above a certain threshold, they are more likely to have a severe condition (e.g., cancer or surgery-related treatment).

Applications and Benefits:

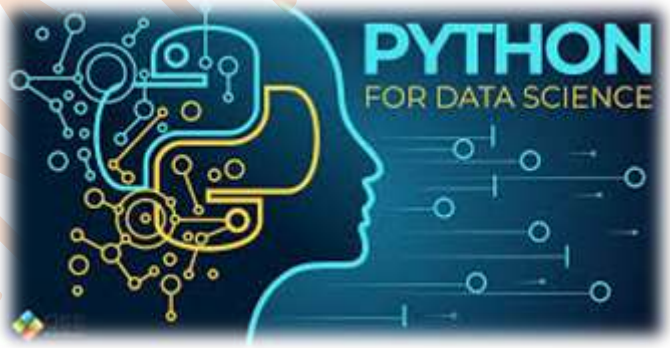
Cost Management: Healthcare providers can use this model to better understand the financial burden associated with different medical conditions. It can help in estimating treatment costs for future patients.

1. **Insurance Pricing**: Insurance companies can use these insights to adjust pricing models for different types of medical conditions, ensuring premiums align with expected treatment costs.
2. **Hospital Resource Planning**: By analyzing which medical conditions tend to have higher costs, hospitals can plan their resource allocation, staffing, and equipment procurement.
3. **Treatment Prediction**: This model could help in predicting treatment types based on a patient's billing amount, guiding doctors and hospital staff in planning care more effectively.
4. **Better Patient Management**: By understanding the relationship between medical conditions and treatment costs, hospitals can manage patient expectations and offer more targeted financial counseling.

Conclusion:

The project demonstrates how to use **ANOVA (Analysis of Variance)** to analyze the relationship between **Billing Amount** and **Medical Condition** in a healthcare dataset. By identifying significant differences in billing amounts across different medical conditions, the project provides valuable insights into healthcare costs and patient conditions. This approach can be extended to real-world healthcare applications, including cost forecasting, resource allocation, and improving the overall financial and operational efficiency of healthcare providers and insurers.

Thank You



Github Link- https://github.com/Somnath342000/Python-ML-project-of-Power-Consumption-Forecasting-state-wise-in-India.git

Github Link- https://github.com/Somnath342000/Python-Unsupervised-ML-Regression-project-on-Fundamental-Analysis.git

Github Link- https://github.com/Somnath342000/Python-Unsupervised-ML-Classification-project-on-Healthcare-Research.git