Problem 1: Car Horsepower Prediction

# Overview:

## Problem Definition:

The objective is to predict car horsepower using a dataset containing information about various car features.

## Dataset Columns:

1. Engine: Numerical value (float) representing car engine capacity.

2. Kilometers\_Driven: Numerical value (float) indicating the number of kilometers the car has been driven.

3. Mileage: Numerical value (float) representing car mileage (Total Distance Travelled / Total Fuel Consumed).

4. New\_Price: Numerical value (integer) representing the price of a new car.

5. Power: Numerical value (float) representing car engine horsepower (target variable).

6. S.No: Numerical value (float) serving as the ID for each car in the dataset.

7. Seats: Numerical value (integer) indicating the number of car seats.

8. Year: Numerical value (float) representing the car's production year.

9. Owner\_Type: Categorical variable (object) representing the number of car owners.

10. Fuel\_Type: Categorical variable (object) indicating car fuel type (petrol / diesel / CNG / LPG / electric).

11. Transmission: Categorical variable (object) indicating car gear transmission (Automatic/Manual).

12. Name: Categorical variable (object) representing car engine type.

13. Location: Categorical variable (object) representing car location relative to the city.

14. Price: Numerical value (integer) representing the price of a new car.

## Methods Used:

We employed seven regression models for car horsepower prediction:

1. Linear Regression: A basic linear model that assumes a linear relationship between features and target variable.

2. Ridge Regression: A regression technique that adds a penalty term to the linear regression objective, preventing overfitting.

3. Lasso Regression: Like Ridge but with a different penalty term, potentially leading to feature selection.

4. Decision Tree Regression: A tree-based model that makes decisions based on feature splits.

5. Random Forest Regression: An ensemble of decision trees for improved accuracy and robustness.

6. Support Vector Regression (SVR): Uses support vector machines for regression tasks, capturing non-linear relationships.

7. Gradient Boosting Regression: An ensemble technique that builds multiple weak models sequentially, correcting errors of the previous ones.

## Mean of Preprocessing and Visualization Steps

1. Handling Missing Values: Identify and address any missing values in the dataset.

2. Label Encoding: Convert categorical variables (Owner\_Type, Fuel\_Type, Transmission, Name, Location) into numerical representations.

3. Feature Scaling: Normalize numerical features to ensure all variables contribute equally to the model.

4. Visualization: Utilize visualizations such as pair plots, histograms, and correlation matrices to explore relationships between variables.

## Preprocessing steps

1. Data Exploration: Initially, examined the first few rows of the dataset (df.head()) to get an overview of the data structure.

2. Conversion of Scientific Notation: Converted the 'New\_Price' column, which initially contained scientific notation, to integer format for better interpretability.

3. Data Types Exploration: Checked the data types of each column (df.dtypes) to understand the nature of features, whether they are numeric or categorical.

4. Data Information Summary: Obtained a concise summary of the dataset using df.info(), providing information on non-null counts and data types for each column.

5. Handling Missing Values: Investigated and addressed missing values using df.isnull().sum(), ensuring the dataset's completeness.

6. Exploration of Unique Values: Explored the number of unique values in each feature using df.nunique(). This helped identify categorical features and assess their cardinality.

7. Numeric and Categorical Columns Separation: Separated numeric and categorical columns for further processing (num\_cols and cat\_cols).

8. Unique Values in Categorical Features: For categorical features, analyzed and reported the unique values in each feature using df[col].nunique().

9. Value Counts for Non-Numeric Features: Examined the distribution of non-numeric features, displaying the top values and their counts to understand the frequency of different categories.

10. Descriptive Statistics: Generated descriptive statistics for numerical features using df.describe(). This provided insights into central tendencies, dispersions, and quantiles.

11. Additional Exploration for Categorical Features: For categorical features, specifically focused on the count and distribution of unique values in each feature.

12. Data Statistics Summary: Summarized the unique values in each feature, emphasizing the count of distinct values to understand the diversity within categorical variables.

## Visualization steps

1. Selection of Numerical Features: Created a list (list\_numerical) containing the names of numerical features by selecting columns with numeric data types.

2. Central Tendency Metrics: Calculated and reported the mean, median, and mode for each numerical feature. This provided insights into the central tendencies of the dataset.

3. Univariate Distribution Visualization: Generated a subplot grid (2 rows, 4 columns) for visualizing the univariate distribution of numerical features using seaborn's distplot. This step helped understand the shape and spread of each feature.

4. Pairplot for Bivariate Analysis: Utilized sns.pairplot to create scatterplots for bivariate analysis, specifically focusing on the relationship between features while differentiating by the 'Transmission' category.

5. Correlation Heatmap: Computed the correlation matrix (corr) to quantify relationships between numerical features. Created a heatmap to visualize the correlation coefficients, enabling identification of potential correlations or multicollinearity.

6. Correlation with Target Feature: Created a heatmap specifically highlighting the correlation between numerical features and the target variable 'Price'. This step helped identify features strongly correlated with the target.

7. Repetition of Correlation Analysis: Repeated the process of creating a correlation heatmap to emphasize the correlation between features and the target variable.

8. Central Tendency Metrics Revisited: Revisited the calculation of mean, median, and mode for numerical features, possibly to observe changes or trends after visualizations.

## Experiment

1. Model Training: Train each regression model using the preprocessed dataset.

2. Evaluation Metrics: Calculate metrics like Mean Absolute Error and Explained Variance Score for each model.

# Result of models

1. Linear Regression:

• Trained the model on the preprocessed dataset.

• Calculated Mean Absolute Error: 11.61.

• Explained Variance Score: 0.899.

2. Ridge Regression:

• Applied Ridge Regression with regularization to prevent overfitting.

• Evaluated performance metrics, including Mean Absolute Error and Explained Variance Score.

3. Lasso Regression:

• Utilized Lasso Regression, considering its feature selection capabilities.

• Evaluated performance using relevant metrics.

4. Decision Tree Regression:

• Trained a Decision Tree model to capture non-linear relationships in the data.

• Calculated Mean Absolute Error: 2.74.

• Explained Variance Score: 0.963.

5. Random Forest Regression:

• Employed Random Forest, an ensemble method, for improved accuracy.

• Achieved Mean Absolute Error: 3.29.

• Explained Variance Score: 0.961.

6. Support Vector Regression (SVR):

• Implemented SVR, leveraging support vector machines for regression.

• Assessed performance metrics.

7. Gradient Boosting Regression:

• Utilized Gradient Boosting Regression for sequential model building.

• Examined model performance using Mean Absolute Error and Explained Variance Score.

# Comparison of Models

• Linear Regression: Demonstrated decent performance with an Explained Variance Score of 0.899 but had a higher Mean Absolute Error compared to other models.

• Ridge and Lasso Regression: Evaluated the impact of regularization on model performance. Ridge addressed multicollinearity, and Lasso potentially performed feature selection.

• Decision Tree Regression: Achieved a low Mean Absolute Error of 2.74 and a high Explained Variance Score of 0.963, indicating strong predictive capabilities.

• Random Forest Regression: Demonstrated robustness and accuracy, with a Mean Absolute Error of 3.29 and an Explained Variance Score of 0.961.

• SVR: Displayed moderate performance with an Explained Variance Score of 0.620 and a Mean Absolute Error of 15.40.

• Gradient Boosting Regression: Offered competitive results, with an Explained Variance Score of 0.937 and a Mean Absolute Error of 8.10.

# Conclusion

In conclusion, the Decision Tree and Random Forest Regression models consistently outperformed other models, showcasing their effectiveness in predicting car horsepower. These models exhibited lower Mean Absolute Error and higher Explained Variance Scores, indicating superior accuracy. Further optimization and fine-tuning may be explored to enhance the performance of the chosen models for deployment in real-world scenarios.

• Decision Tree Regression and Random Forest Regression showed superior performance, with low Mean Absolute Error and high Explained Variance Score.

readiness for modeling. Further optimization and tuning can enhance model performance for real-world deployment.

# Reference:

Dataset: https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho?select=car+data.csv

Problem 2: Diabetes Prediction

# Overview:

## Background:

The prevalence of diabetes has been on the rise, necessitating effective predictive models for early detection. This machine learning project utilizes a dataset containing medical and demographic data to build a classification model capable of predicting whether an individual has diabetes. The goal is to contribute to proactive healthcare by identifying potential cases based on patient parameters.

## Objective:

The primary objective is to develop accurate machine learning models for predicting diabetes. By leveraging features such as age, gender, BMI, hypertension, heart disease, smoking history, HbA1c level, and blood glucose level, the models aim to assist healthcare professionals in identifying individuals at risk of diabetes.

## Dataset:

The Diabetes prediction dataset, obtained from Kaggle here, consists of 100,000 entries with features like age, gender, and medical indicators. The dataset is preprocessed, mapping categorical values to numeric representations for machine learning model compatibility.

# Background:

## Introduction:

The project focuses on utilizing machine learning techniques for diabetes prediction, a critical health concern globally. Leveraging a diverse dataset, the models aim to enhance early detection capabilities, contributing to better healthcare outcomes.

## Data Loading and Cleaning:

The dataset is loaded from a CSV file and undergoes minimal cleaning. Categorical columns, such as 'smoking\_history' and 'gender,' are mapped to numeric values. The dataset is impressively clean, with no missing values.

# Objective:

## Exploratory Data Analysis (EDA):

Visualizations showcase the distribution of diabetes cases, 'HbA1c\_level' counts, and prevalence of 'hypertension' and 'heart\_disease' The absence of strong correlations between columns is noted.

## Machine Learning Models:

Various machine learning models are employed, including Random Forest, Logistic Regression, Decision Tree, SVM, Gaussian Naive Bayes, and KNN. Hyperparameter tuning and feature selection are performed to enhance model performance.

## Pipeline:

A pipeline is implemented to streamline model building and evaluation. Models, including Random Forest, Logistic Regression, Decision Tree, SVM, Gaussian Naive Bayes, and KNN, are standardized using a StandardScaler before training and evaluation.

# Description:

## Methodology:

**Data Preprocessing:**

* Categorical columns are mapped to numeric values.
* Data types are adjusted for compatibility.
* The dataset is confirmed to be clean with no missing values.

**Exploratory Data Analysis (EDA):**

* Visualizations provide insights into the distribution of diabetes cases and key features.

**Machine Learning Models:**

### Random Forest Classifier:

* Achieves a training accuracy of 97.18% and testing accuracy of 97.22%.

### Logistic Regression:

* Achieves a training accuracy of 94.11% and testing accuracy of 93.90%.

### Decision Tree Classifier:

* Achieves a training accuracy of 97.18% and testing accuracy of 97.22%.

1. SVM:

* Achieves a training accuracy of 95.98% and testing accuracy of 95.67%.

### Gaussian Naive Bayes:

* Achieves a training accuracy of 90.42% and testing accuracy of 90.60%.

1. KNN:

* Achieves a training accuracy of 96.41% and testing accuracy of 95.35%.

**Pipeline:**

* Models within the pipeline include Random Forest, Logistic Regression, Decision Tree, SVM, Gaussian Naive Bayes, and KNN.
* Standardization using StandardScaler is applied before training.
* Testing accuracies are evaluated for each model.

## Results:

Random Forest Classifier Accuracy: 97.04%

Logistic Regression Accuracy: 95.92%

Decision Tree Classifier Accuracy: 95.36%

SVM Accuracy: 96.23%

Gaussian Naive Bayes Accuracy: 90.60%

KNN Accuracy: 96.12%

The Random Forest Classifier emerges as the top-performing model with the highest accuracy.

# Conclusion:

The project successfully explores machine learning models for diabetes prediction, demonstrating the effectiveness of various classifiers. The pipeline approach facilitates streamlined model evaluation. The Random Forest Classifier, with its accuracy of 97.04%, stands out as a robust model for diabetes prediction.

# Reference:

Dataset: <https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset>

Problem 3: Data Scientist Job Salaries Analysis

# Overview:

## Problem Statement:

Understanding patterns and clusters within the dataset containing information about data scientist job salaries.

# Dataset Columns:

1. work\_year: The year the salary was paid (integer).

2. experience\_level: Experience level in the job during the year (object).

3. employment\_type: Type of employment for the role (object).

4. job\_title: Role worked in during the year (object).

5. salary: Total gross salary amount paid (integer).

6. salary\_currency: Currency of the salary paid (object).

7. salary\_in\_usd: Salary in USD (integer).

8. employee\_residence: Employee's primary country of residence during the work year (object).

9. remote\_ratio: Overall amount of work done remotely (integer).

10. company\_location: Country of the employer's main office or contracting branch (object).

11. company\_size: Average number of people that worked for the company during the year (object).

# Preprocessing Steps

1. Identified and Handled Missing Values: No missing values were found in the dataset, ensuring a complete and reliable dataset.

2. Removed Redundant Columns: Salary and salary\_currency was removed as salary\_in\_usd is retained for uniformity.

3. Data Transformation: Converted categorical variables to numerical representations using appropriate encoding methods.

# Data Visualization

1. Descriptive Statistics: A summary of key statistical measures was obtained using the describe() function. This provided insights into the central tendency, dispersion, and shape of the salary distribution.

2. Initial Data Exploration: The first few rows of the dataset were examined using head() to gain a quick overview of the data structure.

3. Handling Missing Values: A check for missing values was performed using isnull().sum(). Fortunately, no missing values were found, ensuring the dataset's completeness.

4. Label Encoding: Categorical variables, such as 'experience\_level,' were transformed into numerical representations using label encoding. This step facilitates the application of clustering algorithms.

5. Pair plot: A pair plot was generated using Seaborn to visualize relationships between various numerical variables. This provided a comprehensive view of the dataset's distribution and potential patterns.

6. Histogram of Salary Distribution: A histogram of the salary distribution was created using sns.histplot(df['salary']). This visual representation highlights the frequency of different salary ranges.

7. Elbow Method for Cluster Selection: The Elbow Method was employed to determine an optimal number of clusters for the K-means algorithm. A plot of the within-cluster sum of squares (WCSS) against the number of clusters revealed an elbow point, aiding in cluster selection.

8. 3D Cluster Visualization: A 3D scatter plot was constructed to visualize the clusters formed by the K-means algorithm based on 'experience\_level' and 'salary\_in\_usd'. Clusters were differentiated by color, and centroids were highlighted in yellow. This visualization enhances the understanding of how data points group together.

# Model Implementation

1. K-means Clustering:

• Applied the K-means algorithm to group data points based on similarity in experience levels and salary in USD.

2. Hierarchical Clustering:

• Utilized hierarchical clustering to identify hierarchical relationships within the dataset.

# Results Evaluation

1. Silhouette Score: 0.3104

• A measure of how well-defined the clusters are. Higher values indicate better-defined clusters.

2. Calinski-Harabasz Index: 1667.30

• Indicates the ratio of between-cluster variance to within-cluster variance. Higher values signify better-defined clusters.

3. Davies-Bouldin Index: 1.1999

• Measures the compactness and separation between clusters. Lower values suggest better clustering.

# Conclusion

The analysis successfully identified distinct clusters within the data scientist job salary dataset based on experience levels and salary in USD. The chosen clustering algorithms (K-means and hierarchical) yielded promising results, as evidenced by the evaluation metrics. The absence of missing values and careful preprocessing ensured the reliability of the findings. The visualizations provide an intuitive understanding of the relationships and patterns within the data. This analysis lays the foundation for further exploration and decision-making related to data scientist job salaries.

Problem 4: Chest X-ray Image Classification using Vision Transformer (ViT)

# Overview

## Background:

Medical image analysis plays a crucial role in diagnosing and monitoring various diseases. Chest X-rays are commonly used to detect respiratory conditions such as pneumonia and, more recently, COVID-19. This machine learning project focuses on classifying chest X-ray images into three categories: normal, COVID-19, and pneumonia.

## Objective:

The main objective is to develop a deep learning model that can accurately classify chest X-ray images, aiding healthcare professionals in the early detection of respiratory diseases.

# Dataset:

## Description:

The dataset comprises chest X-ray images divided into three classes - normal, COVID-19, and pneumonia. The images are preprocessed using a pipeline that involves reading, resizing, and normalization. Additionally, data augmentation techniques such as random brightness and contrast adjustments are applied to enhance the model's robustness.

## Data Preprocessing:

The preprocessing pipeline involves the following steps:

1. Reading: Images are read using TensorFlow's IO functions.

2. Resizing: Images are resized to a standard size (IMG\_SIZE) to ensure consistency.

3. Normalization: Pixel values are scaled between 0 and 1, converting images to the float32 data type.

# Methodology:

## Model Architecture:

The chosen architecture for this project is the Vision Transformer (ViT). ViT has shown promising results in image classification tasks by utilizing self-attention mechanisms to capture long-range dependencies in the data.

## Model Details:

- ViT model (vit\_b16) is employed as the base architecture for feature extraction.

- Additional fully connected layers are added for further abstraction and classification.

- Activation functions used include GELU (Gaussian Error Linear Unit) for non-linearity.

- The final layer employs the softmax activation function for three-class classification.

## Training Configuration:

- AdamW optimizer is utilized with a learning rate of 0.0001 and weight decay of 0.001.

- Sparse categorical cross-entropy loss is chosen for the multi-class classification task.

- Model performance is evaluated based on accuracy.

# Results:

## Model Summary:

The summary of the implemented model is as follows:

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) [(None, IMG\_SIZE, IMG\_SIZE 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

vit (Functional) (None, 16, 16, 768) 85798688

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 196608) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 256) 50331648

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 64) 16448

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 32) 2080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 3) 99

=================================================================

Total params: 136,122,963

Trainable params: 136,122,963

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Experiment

## Training:

The training process involved training the ViT-based chest X-ray classification model for 5 epochs. Each epoch consists of 161 batches, and the model's performance was evaluated at the end of each epoch. The following are the training results:

### Epoch 1/5:

- Training Loss: 0.2570

- Training Accuracy: 90.01%

- Time per Epoch: 253s

### Epoch 2/5:

- Training Loss: 0.1288

- Training Accuracy: 95.39%

- Time per Epoch: 207s

### Epoch 3/5:

- Training Loss: 0.0809

- Training Accuracy: 97.01%

- Time per Epoch: 207s

### Epoch 4/5:

- Training Loss: 0.0778

- Training Accuracy: 97.53%

- Time per Epoch: 207s

### Epoch 5/5:

- Training Loss: 0.0616

- Training Accuracy: 98.06%

- Time per Epoch: 214s

## Testing:

After training, the model was evaluated on a separate test dataset to assess its generalization performance. The ViT model achieved the following results on the test set:

# ViT Model Test Results:

- Test Loss: 0.296

- Test Accuracy: 91.149%

- Evaluation Time: 22s

# Discussion:

The training results show a steady decrease in the training loss and an increase in training accuracy over the five epochs, indicating that the model is learning and improving its performance on the training data.

The test results reveal a high accuracy of 91.149%, demonstrating the model's ability to generalize well to unseen data. The performance on the test set suggests that the model has successfully learned meaningful features from the chest X-ray images and can effectively classify them into the three specified classes.

The model's training time per epoch is reasonable, considering the complexity of the ViT architecture and the size of the dataset. Further experimentation, such as hyperparameter tuning or exploring different architectures, could potentially improve the model's performance or efficiency.

In conclusion, the experiment demonstrates the effectiveness of the ViT-based model for chest X-ray image classification, highlighting its potential for aiding medical professionals in diagnosing respiratory conditions.

## Reference:

Dataset: https://www.kaggle.com/datasets/alsaniipe/chest-x-ray-image