Project Approach

We started the project by acquiring the apt data set. We imported some libraries like Pandas, NumPy, Matplotlib, Pickle, Warnings, and Scikit-learn. Pandas is used for data manipulation and analysis, NumPy library is a fundamental package for scientific computing in Python. The matplotlib library is a powerful plotting and visualization tool in Python. It is used to create a wide range of static, animated, and interactive visualizations, including line plots, scatter plots, histograms, bar charts, and more.The pickle module is used for object serialization and deserialization.The seaborn library is built on top of Matplotlib and provides a high-level interface for creating attractive and informative statistical graphics.The warnings module is a Python standard library used to handle warning messages.Finally, Scikit-learn, also known as sklearn is a comprehensive machine learning toolkit in Python.

The dataset we chose had a lot of garbage values within it therefore, we used Microsoft Excel to split the columns and cleaned the data by removing garbage values. The price column already had numerical values, but we had to remove the commas from it. No duplicate values were found.

By using the data.isna().sum() we calculated the number of null values in each column of the DataFrame 'data'. Furthermore we dropped columns fast_charge, card, card_size, refresh_rate, displaypx_h, displaypx_w, rear_cam_2, rear_cam_3, front_cam_2, 3G and rating. We remove rows that have missing values in the "rom" and "front_cam_1" column. We filled the missing value in the clock_speed, core, battery columns with the mean value of the column. The missing values in 4G and 5G columns are filled with zero.

After performing correlation analysis using corr = data.corr(), sim count, display size(inches), cores, clock speed, front and rear camera information, RAM, ROM, 4G, 5G, battery and OS were used to train the machine learning models for predicting the price of a mobile phone. We also used other visualizations like Boxplot to understand data distribution through quartiles. Plotting the lineplot of ROM and RAM size against price helped us come to the conclusion that as the ROM and RAM size increases price also increases. We created data visualizations using pie charts to see the distribution of OS, and 5g. We came to the conclusion that iphone varieties are less compared to android phones in the dataset and that half the observations in the data have 5G.

We created four separate datasets for training and testing. 'x_train' contains independent variables for model training, 'y_train' has the corresponding target variable for the training

data. ‘x_test’ has independent variables for model evaluation and ‘y_test’ has corresponding target variable for testing data.

We trained three different machine learning models, Linear Regression, Random Forest Regression and Decision Tree model on the data. First the Linear regression is called on the x_train and y_train. The linear regression model learns the coefficients for each feature to approximate the relationship between the independent variables and the target variable. Random forest model consists of multiple decision trees, and each tree is trained on a random subset of the training data. It learns to make predictions by aggregating the predictions of individual trees. DecisionTreeRegressor().fit() fits the decision tree regression model to the training data. The decision tree model learns a hierarchical structure of binary splits based on the features in the training data, aiming to minimize the mean squared error in predicting the target variable.

Performance of the machine learning models are evaluated using two metrics, training and testing scores. A higher training score indicates a better fit of the model to the training data, and higher testing score indicates better fit of the model to the testing data. Values of these range between 0 and 1. We want a training score close to the testing score, it suggests that the model is performing well and generalizing adequately to unseen data.

Training and Testing scores for the models are :

- Linear regression:
  Training : 0.701065139742171
  Testing : -39.26283868118925
- Random Forest:
  Training : 0.9721126158373805
  Testing : 0.8658217900727673
- Decision Tree:
  Training : 0.9981244520875332
  Testing : 0.7909795540584875

We realize from the above results that the training and testing scores of the Random Forest model is very close and hence it is a well performing model.

The unknown data point extracted from a newly released mobile’s features from Flipkart is passed to the model for prediction. This involves converting the feature data (a) into a NumPy array and expanding its dimensions to match the expected input shape of the model. The trained model is then used to predict the price for the given feature data.

The predicted price is printed to provide the final result. It is common practice to round the prediction to an appropriate number of decimal places to enhance readability and presentation.
The real price of the Smartphone from Flipkart is 18490 and the predicted value is 18546,which came out to be pretty accurate.

Used the cross_val_score function to perform cross-validation on the Random Forest model.Provided the model (rf), input features (x), and target variable (y) as arguments.Set the number of cross-validation folds (cv) to 10. This means the data will be split into 10 equal parts, and the model will be trained and evaluated 10 times, with each part serving as the test set once.Specify the evaluation metric (scoring) as needed. ‘’R2' can be used to calculate the coefficient of determination (R^2 score) for each fold.

The cross_val_score function returns an array of scores, one for each fold of cross-validation.Print these scores to assess the model's performance on different subsets of the data.

The result of 10-fold cross-validation provides an evaluation of the Random Forest model's performance across multiple subsets of the mobile phone dataset. It helps to understand how well the model generalizes and provides insights into its overall effectiveness for price prediction.

Pickling allows us to serialize the model and save it to a file, enabling us to load and utilize the model at a later time.

Pickling the Random Forest model:The Random Forest model (rf) is pickled using the pickle.dump() function.This function takes two arguments: the model object (rf) and a file object created with the open() function in binary write mode(“wb”).The pickled model is saved into a file named "randomforest.p" using the open() function.

Pickling the Linear Regression model:The Linear Regression model (lr) is pickled using the same process as the Random Forest model.The model is pickled using pickle.dump(lr, open("LinearRegression.p", "wb")).The pickled model is saved into a file named "LinearRegression.p".

Pickling the Decision Tree model:The Decision Tree model (dt) is pickled using the same process as the previous models.The model is pickled using pickle.dump(dt, open("DecisionTree.p", "wb")).The pickled model is saved into a file named "DecisionTree.p".

By pickling the models, we can store them for future use without the need to retrain them. This allows for easy transportation, deployment, and utilization of the trained models in various applications. The pickled files can be loaded at any time to make predictions or perform further analysis using the trained models.

After saving the model,we decided to containerize our application using docker.Docker simplifies application development, deployment, and runtime by using containerization. It provides isolation, portability, efficient resource utilization, fast deployments, scalability, version control, simplified dependency management, and collaboration. Overall, Docker streamlines processes, improves portability, and provides a consistent runtime environment.

We created a Dockerfile for dockerizing the main jupyter notebook file.We used the base image of jupyter/base-notebook.The main jupyter notebook file(mobileprice.ipynb),the data file used(final.csv) and the python module requirements specification file(requirements.txt) was copied to the app directory in the docker image.All the python modules mentioned in the requirements file was installed and we exposed the port 8888 for running the jupyter notebook. The command for the image while running the container was set to "jupyter notebook --ip=0.0.0.0 --no-browser --allow-root".

The docker image was created using the command “docker build -t ghcr.io/k-nath79/mobile-price-prediction:latest .”. After that the image was pushed to the github container registry and we connected this package to our github repository.

This image can be pulled from the github container registry using the command “docker pull ghcr.io/k-nath79/mobile-price-prediction:latest”.To run the image using docker in an interactive terminal we can use the command “docker run -ti -p 8888:8888 ghcr.io/k-nath79/mobile-price-prediction:latest” where we bind the port 8888 of the container to port 8888 of the host machine.It will open up a jupyter notebook environment in the browser where we can run our main jupyter notebook file directly without the hassle of installing all the packages.

In Conclusion,The machine learning models were applied to predict the price of mobile phones using the provided dataset. Linear Regression, Random Forest, and Decision Tree models were utilized. The Random Forest model exhibited the highest performance, followed by Linear Regression and Decision Tree. Cross-validation scores were also calculated, showing the models' consistency. The trained models were saved for future use. Overall, the Random Forest model is recommended for accurate price prediction in mobile phones also the Docker Image file was created for easy access.