TIEVS – DYNAMIC CAR PRICE PREDICTION SYSTEM USING ENSEMBLE LEARNING

K.S.S Bandaranayake

(IT18113532)

BSc (Hons) in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

TIEVS – DYNAMIC CAR PRICE PREDICTION SYSTEM USING ENSEMBLE LEARNING

K.S.S Bandaranayake

(IT18113532)

Dissertation Submitted in Partial Fulfillment of the Requirements for the BSc (Hons) in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

DECLARATION

I declare that this is my own work, and this dissertation does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or institute of higher learning, and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name	Student ID	Signature
K.S.S Bandaranayake	IT18113532	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: Date:

Abstract

The used car population has been growing exponentially at an increasing rate. In contrast, car resellers occasionally find it tedious to finalize a realistic price according to demand, making buyers insecure. Presently reselling cars are advertised through online classifieds. Therefore classified platforms need their own independent price prediction mechanism rather than third-party price crawlers. Ergo, this study prominently focused on implementing a dynamic used car price prediction system integrated with 'Tievs' classified advertising platform to provide the customers with a convenient and fair price value. A complex 'Craiglist' used car dataset was utilized, with approximately 450,000 records with 26 distinct car attributes, condensed and appropriately transformed after scrupulous data engineering and exploratory data analysis. Next, a comparative study on 15 different regressor models was done by continuous training and testing from the transformed dataset. Extra Trees, Bagging, Random Forest, and Gradient Boosting regressors became rare top winners with Adjusted R-squared values being 0.972, 0.956, 0.920, and 0.872, respectively. K-fold cross-validation, RandomizedSearchCV, and GridSearchCV were used to reduce model overfitting and enhance generalization. Chosen regressors were combined using the stacked generalization regressor ensemble to harness multiple prediction strengths towards a formidable hybrid model, having bolstered accuracy, efficiency, and reliability with excellent performance than individual regressors. The ensemble model was integrated into an Application Programming Interface built from Quart Python framework, which provided a POST endpoint to predict, communicate, and visualize price through the Angular framework implemented client application. All elements were finally deployed in a purchased cloud server from DigitalOcean to promote admiring application scalability and availability. This inaugurated novel methodology replenished a significant research gap within the forecasting domain. Tuning hyperparameters, analysis of more sophisticated algorithms, and expansion of the price prediction to other types of classified items could be prospective future avenues for this research.

Key Words – Classified Advertising, Ensemble Learning, Machine Learning, Regressor Comparative Study, Stacked Generalization, Used Car Price Prediction

ACKNOWLEDGEMENT

Humble gratitude is bestowed upon this research's supervisor, Ms. Manori Gamage (Lateral Entry Coordinator, Senior Lecturer - Faculty of Computing) and co-supervisor Suriyaa Kumari (Lecturer – Faculty of Computing) for the continuous invaluable assistance, support, guidance, advice, and motivation provided throughout the whole one year scoped research. The feedback and suggestions given were immensely precious and effective to attain the set goals and objectives of this research and commute in a smooth path towards successful completion. The supervisors' eagerness and enthusiasm towards both miniature and cumbersome research processes, motivated quality and efficient methodologies to be implemented and build up the research potential and capability parallel to competition level which otherwise would have been impossible to achieve.

Furthermore, special thanks are implicitly given to Ms. Wasanthi Bandaranayake for sharing her positive and negative experiences with online classified advertising portals and presumptive enhancements that should have been incorporated within an advanced technological aspect. Her ideas and propositions opened up a novel research avenue that is yet to be explored and investigated among existing literature relating to information technology and business fields.

Additionally, since Mr. Austin Reese and the entire Kaggle team produced and maintained the evaluation-rudimental dataset that contained the online advertised used car classified records scraped from Craiglist, wholehearted appreciation is extended to them as well. These open datasets were made freely available in order to facilitate numerous experimentations conducted by many researchers, including this own study.

Moreover, heartfelt thanks and respect are inferred upon parents to acknowledge the fortitude, effort, and time provided by them when acquiring the essential resources and settling other spontaneous expenditures. Ultimately, encouragement and collaboration obtained from the research group members regarding every milestone in this initiative for favorable outcomes are gratefully treasured.

TABLE OF CONTENTS

Declaration i

Abstract ii

Acknowledgement iii

Table of Contents iv

List of Figures vi

List of Tables ix

List of Abbreviations x

List of Appendices xi

1 Introduction 1

1.1 Background 1

1.2 Background Literature Survey 6

1.3 Research Gap 10

1.4 Research Problem 13

1.5 Research Objectives 18

2 Methodology 20

2.1 Methodology 20

2.1.1 Theoretical research methodology 20

2.1.2 High-level system design 24

2.1.3 Software development life cycle 27

2.1.4 Technologies and tools utilized 27

2.1.5 Project requirements 28

2.2 Commercialization Aspect of the Product 29

2.3 Implementation 30

2.3.1 Dataset realization 30

2.3.2 Data preprocessing & exploratory data analysis 32

2.3.3 Comparative study on 15 regressor models 41
2.3.4 Top 4 regressor infused ensemble model development 48
2.3.5 Ensemble model serialization 52
2.3.6 Flask API implementation & model deserialization 53
2.3.7 API & model deployment in cloud server 55
2.3.8 Front-end application development, integration & deployment 59
2.4 Testing 62
3 Results & Discussion 66
3.1 Results 66
3.2 Research Findings 76
3.3 Discussion 79
3.4 Summary of Student's Contribution 83
4 Conclusion 84
References 88
Appendices 93
Appendix A: Price Prediction System Test Cases 93
Appendix B: Tievs User Feedback Questionnaire 95
Appendix C: Turnitin Plagiarism Report 99
Appendix D: Turnitin Plagiarism Report Receipt 102

LIST OF FIGURES

Page

Figure 1.1 Number of Registered Cars from 2003 to 2014 [1] 2
Figure 1.2 Vehicles Operating in US from 1991 to 2019 [2] 3
Figure 1.3 Projection of Chinese Vehicle Growth [3] 3
Figure 1.4 New and Used Light-Duty Vehicle Sales in US [4] 4
Figure 1.5 Unrealistic car classified price part 1 13
Figure 1.6 Unrealistic car classified price part 2 14
Figure 1.7 Main types of ensemble learning 17
Figure 2.1 Research methodology diagram 20
Figure 2.2 Tievs price prediction system overview diagram 24
Figure 2.3 Commercialization aspect of the product 29
Figure 2.4 Dataset first five rows 30
Figure 2.5 Dataset Kaggle view 31
Figure 2.6 Dataset columns 31
Figure 2.7 Insignificant column removal 33
Figure 2.8 Correlation heatmap between numerical variables 34
Figure 2.9 Numerical column null value handling 35
Figure 2.10 Categorical column null value handling 35
Figure 2.11 All null values handled 36
Figure 2.12 Mean prices by manufacturer 36
Figure 2.13 Price range normalization 37
Figure 2.14 New price distribution 37
Figure 2.15 Number of cars by manufacturer 38
Figure 2.16 Selection of records pertaining chosen car manufacturers 38
Figure 2.17 Car model filtration 38
Figure 2.18 Year range refinement 38
Figure 2.19 Higher car make years 39
Figure 2.20 Condition attribute tinkering code 39
Figure 2.21 Distribution of cars by condition 40
Figure 2.22 Code for One-Hot-Encoding 41

Figure 2.23 One-Hot-Encoded categorical columns 41
Figure 2.24 Cleaned dataset export 41
Figure 2.25 Dataset train:test split 41
Figure 2.26 Feature Scaling 45
Figure 2.27 GridSearchCV python code 45
Figure 2.28 RandomizedSearchCV python code 45
Figure 2.29 Specifically created arrays and functions 46
Figure 2.30 Model accuracy output visualization function 47
Figure 2.31 Random Forest bagging structure [35] 50
Figure 2.32 Gradient Boosting ensemble structure [35] 50
Figure 2.33 Stacking algorithm [36] 51
Figure 2.34 Stacking regressor custom functions 52
Figure 2.35 Pickle model dump 53
Figure 2.36 Pickle model column dump 53
Figure 2.37 Flask API part 1 54
Figure 2.38 Flask API part 2 55
Figure 2.39 DigitalOcean droplet dashboard 57
Figure 2.40 Specific droplet dashboard UI 57
Figure 2.41 Ubuntu server SSH tunnel 58
Figure 2.42 Status of the Supervisorctl 58
Figure 2.43 Cloudflare Tievs dashboard part 1 60
Figure 2.44 Cloudflare Tievs dashboard part 2 60
Figure 2.45 Communication from prediction model to customer & vice versa 61
Figure 2.46 Postman price prediction API test 63
Figure 2.47 Price prediction Loadmill test part 1 65
Figure 2.48 Price prediction Loadmill test part 2 65
Figure 3.1 Python code for comparison table generation 68
Figure 3.2 Regressor comparison table 68
Figure 3.3 Adjusted R-squared value calculation code 70
Figure 3.4 Top 4 regressor predicted values vs. actual values 71
Figure 3.5 Individual and ensemble model accuracy comparison 72
Figure 3.6 Individual and ensemble model mean score boxplot comparison 73

Figure 3.7 Front-end application home page 74
Figure 3.8 Predicted price visualization in ad submission form 74
Figure 3.9 Tievs efficiency report from GTmetrix part 1 75
Figure 3.10 Tievs efficiency report from GTmetrix part 2 75
Figure 3.11 Research-Conducted Machine Overview 80

LIST OF TABLES

Page

Table 1.1 Comparison of key previous researches 12
Table 2.1 Descriptive statistics of categorical variables 32
Table 2.2 Descriptive statistics of numerical variables 32
Table 2.3 Comparatively studied regression algorithm models 42
Table 3.1 Top 4 regressor model optimal hyperparameters 69
Table 3.2 Summary of personal contribution 83

LIST OF ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
CORS	Cross Origin Resource Sharing
CPU	Central Processing Unit
IT	Information Technology
KNN	K-Nearest Neighbors
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Standard Error
RF	Random Forest
RMSE	Root Mean Square Error
RSE	Relative Squared Error
SVM	Support Vector Machine
UI	User Interface
URL	Uniform Resource Locator
WWW	World Wide Web

LIST OF APPENDICES

Appendix	Description	Page
Appendix – A	Price Prediction System Test Cases	93
Appendix – B	Tievs User Feedback Questionnaire	95
Appendix – C	Turnitin Plagiarism Report	99
Appendix – D	Turnitin Plagiarism Report Receipt	102

1 INTRODUCTION

1.1 Background

With information technology developments, the usage of online classified advertising portals has been growing immensely throughout recent years due to many factors such as the higher availability, accessibility, and less physical interactions than traditional classified deliverances (newspapers, magazines, booklets, leaflets tend to get inaccessible during a lockdown) [1], ease of use, distributed-architectures, economic convenience, less consummation of precious time out of people's busy schedules and having access to a wider range of customers and sellers. Some of the prominent examples for present classified advertising portals around the world are Quikr, Olx, Craiglist, Carewale, CarDheko, ikman.lk, Facebook marketplace, etc.

Nevertheless, some scenarios can occur when customers are not quite capable of finalizing a selling price value for the product that they hope to market and presumably will continue to perform extensive research on how to perceive and clarify a satisfactory price value. When engaged in price clarification, a huge amount of effort must be put into the process, all the while sacrificing valuable time yet again, and in doing so, the customers might feel rather lethargic to continue on with the sale of the product altogether. This withdrawal idea will consequently result in a degradation of satisfying the customer's main requirements and necessities delivered by the particular classified advertising application. As a result, the portal may lose its maintained reputation and competitiveness with other similar systems as well. Although our research project's ultimate goal is launching the Tievs application after thorough designing and developments to cater to numerous types of classifieds in the future, along with more upgrades relevant to each criterion, on behalf of the scope of this project that is to be completed within 12 months, we have decided on restricting our research implementations, with relevance to only 'Car' classifieds. With that being stated, this thesis will be emphasizing on the analysis and investigations done

regarding the used car price prediction functionality that finally should be integrated with the Tievs application, along with the rest of the member researched components.

Redirecting the attention, the population of used or second-hand cars in the world market has been increasing ever since and might have been doubled in the last few years, primarily due to the Covid-19 pandemic. Since the traditional ways of trading modes have been unable to meet the demands of the consumer, the online trading platform succession trend has been inevitable for the resale of vehicles altogether.

Initially, for brand new cars, the price is incurred by its manufacturer and the government, in addition to various taxing schemes, which then leads to a higher price than most people cannot effortlessly afford due to insignificant amount of funds procured, resulting in a global increase of used and private cars, which are comparatively modestly priced and has become the first choice of most car buyers. According to data gathered and received from National Transport Authority in 2014 [1], there has been a 254% increase in the number of cars from 2003 to 2014, as shown in Fig. 1.1, which is a tremendous variation itself, from 68,524 to 160,701. It infers that brand new vehicles only acclaim a limited space, out of all cars sold each and every year.

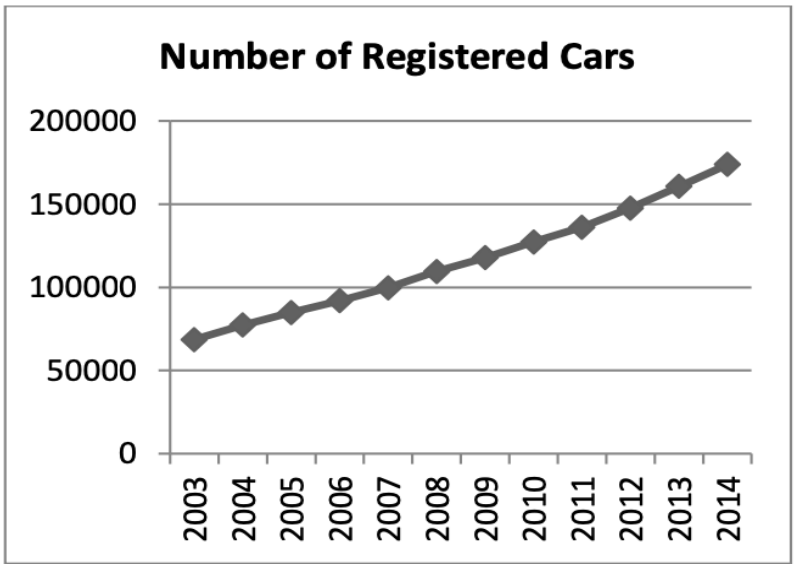


Figure 1.1 Number of Registered Cars from 2003 to 2014 [1]

As shown in Fig. 1.2, which was acquired from WOLF STREET [2], in addition to Fig. 1.3, which was obtained from 'ACCESS' online magazine [3], when we carefully analyze the data of the number of passenger vehicles currently in operation in the United States, and the projection of the cars in China in the past years and upcoming years, it is plausible to think that the worldwide trend takes a similar form and has continued unceasingly, and in an increasing rate at that as well, regardless of the "Great Recession" period shown in Figure 1.2.

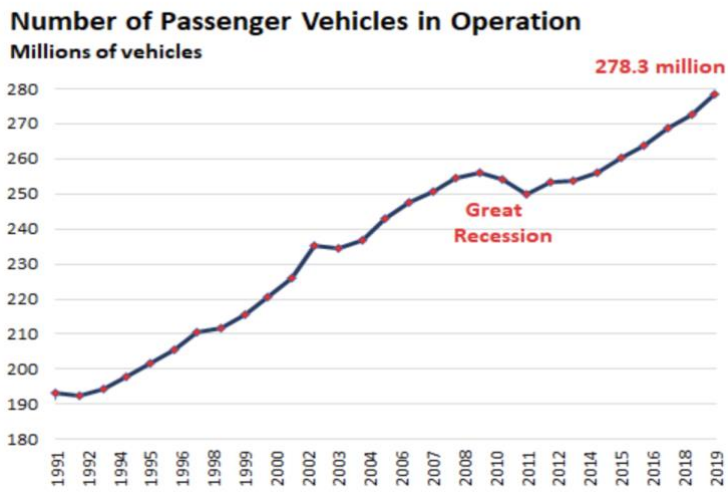


Figure 1.2 Vehicles Operating in US from 1991 to 2019 [2]

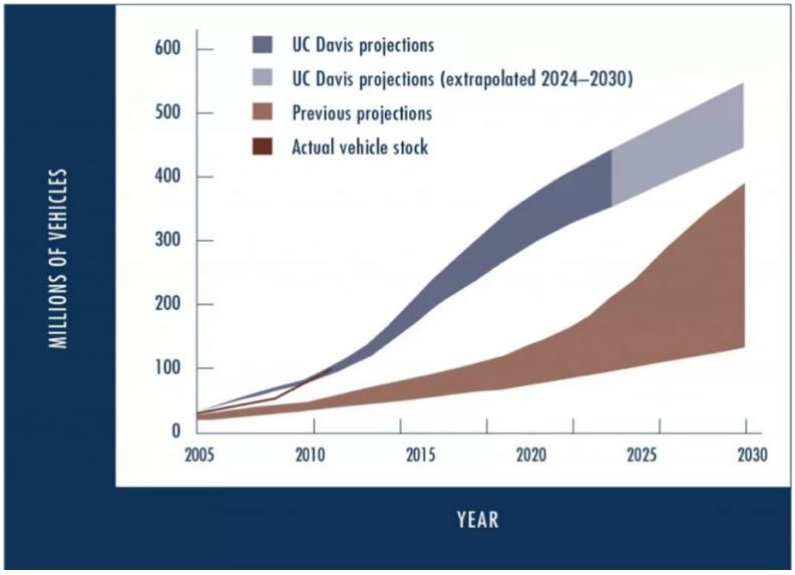


Figure 1.3 Projection of Chinese Vehicle Growth [3]

The used light-duty vehicle market has progressed to be more than twice the size of new light-duty vehicles within the US, as the statistic variation stands at being 23 million between the two types [4], which Fig. 1.4 clearly depicts as well.

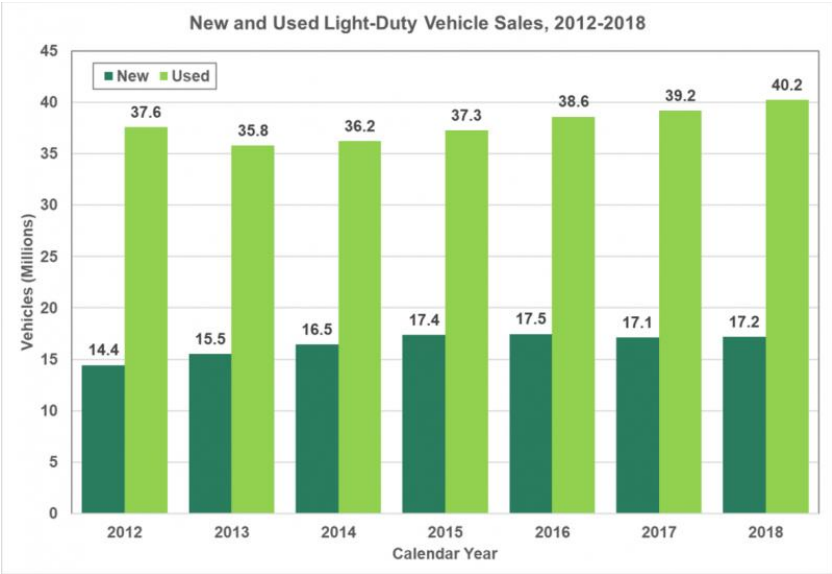


Figure 1.4 New and Used Light-Duty Vehicle Sales in US [4]

As evidence confirms, the fact that second-hand cars within the market have soared is undeniable, and successive reconstruction mechanisms followed by vehicle repairers may have contributed modestly towards it. However, many cars at present are bought on a lease basis, and after the seller and buyer contract is completed, the buyer is able to resell the car at his or her own discretion. As customers can be lured into impractical prices and fall into misconceptions, it is affirmative that an effective and efficient used car price prediction system is an undeniable necessity to establish accurate worthiness for a car without any unnecessary inconveniences. Moreover, the preciseness of second-hand car price evaluation highly dictates whether both buyer and seller can receive a more productive trading ordeal. Hence, it is why that our Tievs application should include implementation of an internal process to predict the price of a product, that a certain customer might decide on advertising, and suggest the forecasted price value as a suggestion so that the customer may have the privilege of either accepting and proceeding further with it or simply ignoring it. In this manner, the customer would be more appreciative and eager to stay and utilize the application expedite,

without disconnecting from it, due to a single but troublesome matter such as finding out the price value themselves. As important and seamless as it is, the predicting process of the resale prices of the used or second-hand cars is a tedious and challenging task to undergo, with a high accuracy level, when considering the variety of features and characteristics a car might possess, that promotes deviations in the resale price. Some of the examples are manufactured year, model, body type, mileage, transmission, braking system, interior type, cylinder volume, fuel max liter capacity, number of doors, size of the car, number of wheels, steering wheel, air conditioner, engine capacity, GPS navigators, overall status, manufacturer awards, car color, braking system, interior designs, acceleration, safety index, car weight, previous owners and types (if the car had gone through any accidents or was it a lady-driven car), and due to the rise of the fuel prices all around the world, the fuel economy and the type of fuel plays a major role in predicting a price for a second-hand car as well, although most people are not aware of the actual amount of fuel that is burned for each km driven by their car. Unfortunately, more or less of these features are considered when people tend to buy a vehicle, and certainly, only a few of them directly and largely affect the price value.

In this component, more impactful characteristics of cars will be considered. The prediction will be made by using the four most optimal Machine Learning (ML) models developed out of 15 different regression-based algorithms, through thorough finetuning hyperparameters and training the models, which then will ultimately result in an ensemble model type, while using the 'Used Cars Dataset' obtained from Kaggle, which contains prices and attributes of 450,000 of US reselling car details that are updated within the past three months by Austin Reese and extracted from the online classified advertising platform 'Craiglist' by using a web scraper that monitors for data monthly. The dataset will ultimately be preprocessed, and only the relevant columns that make a contribution to the variation for the selling price will be taken into account. Each algorithm's respective achievements and performance will be analyzed concerning the dataset, and the best modal having an inconsequential absolute mean absolute error and other considerable parameters having an exemplary value will be chosen to be used and integrated along with the Tievs application.

1.2 Background Literature Survey

Scientific literature focused on existing classified platform price valuation has been obscure. Therefore this section will be debating on several types of research that have been conducted in recent years regarding used car price prediction, where some were phenomenal, revealing results about the accuracy and efficiency levels of ML algorithms that have been used for prediction.

Ganesh et al. [5] selected Multiple Regression, Lasso Regression, and Regression Trees algorithms to produce predictive models and used One-way Analysis Of Variance with Post-Hoc test to determine error rate variances which were quite avant-garde. Lasso and Regression Trees were deemed better, with error rates less than 5%. However, their results would have demeaned if a larger data sample was used to train and test than using only 804 records with 11 decisive features, being quite similar to Kuiper [6], who used a Multivariate regression model to predict numerical price values for 2005 General Motor cars using the same dataset extracted from a famous Pakistani car reselling online platform called PakWheels, encouraging the worth of variable selection techniques and analyzing multiple models' performances.

Noor and Jan [7] used a supervised ML technique, a multiple linear regression model, to predict the price of a car, and their dataset was collected during a two-month period, having car attributes such as model, cubic capacity, mileage, manufactured year, transmission type, number of ad views, brand, exterior type, power steering, rims type, engine type, city, registered city, version, date when the ad was posted, and price. They collected 2000 car data records from PakWheels, having the prices of them tagged, and preprocessed the data, which was then finalized to 1699 records. The authors applied variable selection techniques to find only the most relevant features from the list, such as model, model year, price, and engine type. Since regression is based on numerical data, text values in columns as engine type and model were converted to codes, i.e., 1,0, and then fed into the model using Minitab. In conclusion, the authors were able to achieve an impressive R-sq value of 98.61% and R-sq (adj) value of 98.50%, since depending only on R-sq is impractical due to natural variability when

adding more predictors into the model and contaminate the pure accuracy by making R-sq value higher without solid verifications. Nevertheless, their approach was unreliable since only 1699 records and three features were considered from feature engineering, reducing forecast perfection and permitting overfitting.

In [8], a comparative study examined 3 regression-based models where the Gradient Boosting model gave an admirable accuracy having mean absolute error as 0.28 in comparison to Random Forest (RF) and Multiple Linear Regression. Their staggering 304,133 data records with 11 predictors were applauding for model efficiency. Regardless, ensemble techniques yielding better performances were not explored in theirs, unlike this research. One study [9] found RF classifier the best out of 6 others, having test accuracy of 83.08% with only 5 relevant features and withdrawing main ones as vehicle type, price, and transmission, surfacing a feature selection flaw with low broadened ranges, as opposed to ours with 13 estimators having adequate ranges. Pal et al. [10] investigated Linear Regression and RF, with 3 notable but ill-suited forecasters, and RF triumphantly conferred train and test accuracies as 95.82% and 83.63%. Overfitting still seemed to linger in contemplation of the above values and despite cross-validation and grid search measures taken.

Authors [5]-[10] utilized datasets scraped from e-commerce websites and/or from Kaggle. Hence they originate the notion that peculiar car price knowledge is futile and confirms the sufficiency of open datasets that inspired this study as well.

Contradictorily, Pudaruth [11] has predicted the price for Toyota and Nissan cars in Mauritius by manually collecting the necessary historical data from daily newspapers in less than a month's period since time factors also play an important role in the car price. Attributes like the model, cubic capacity, mileage, production year, exterior color, brand, transmission type, and price of a car were studied by him. Four different techniques like multiple linear regression, decision trees (Random Forest and J48, which is a Java implementation of C4.5), k-nearest neighbors, and Naïve Bayes were compared by the author to finding out the best model. The data were normalized to prevent large values when used with KNN, and numerical values were converted to

nominal values to be used with decision tree algorithms. It was concluded that Naïve Bayes and decision tree methods were unable to handle numerical output classes while KNN obtained the best results having a mean absolute error of 27,000 for Nissan cars and 45,000 for Toyota cars. Due to the limited number of data records (97) considered, the performance accuracy was not at an excellent level, i.e., being less than 70%, which can be seen as a major flaw of the research methodology.

Similar to [11], Peerun et al. [12] conducted research on predicting the price of Mauritian reselling cars by comparing the performance of four ML algorithms such as SVM, KNN, Linear Regression, and MLP (cycles = 5000, learning rate = 0.5). They have gathered 200 record dataset through car websites and daily newspapers, during an interval of one month (month of August 2014), having car features as the year, make, paint, engine, mileage, and transmission type, along with the specific prices given. The authors' experiment results depicted that SVM and neural network regression (MLP) had slightly better performance than linear regression when comparing the accumulated MAE (in Rupees) values which are, respectively, 30605 for SVM, 30746 MLP, 30828 for linear regression. KNN had the largest MAE, which was 42240, and was considered the worst accurate of the four approaches. Even though the MAE values are higher for these algorithms than the values obtained by Pudaruth [5], the authors have considered a higher amount of records (200) as opposed to 97 records. They have considered six predictor factors than just three factors, making the prediction somewhat satisfactory since the deviation is only less than 10% from the actual price, though not exceptional due to the higher gap between predicted and actual price values.

As described by Sun et al. [13], the optimized algorithm they have built, called the Like Block-Monte Carlo Method (LB-MCM), by basing BP neural network algorithm using 1630 records of data, identifies hidden neurons more swiftly, having both generalization and approximation ability. The authors infer that the built model will increase convergence speed, accuracy, and network topology than traditional formula methods while providing price prediction for a P2P car trading system. The absolute and relative error rates encountered prior to tweaking the BP neural network algorithm

was respectively 0.113+/-0.080 and 0.78%+/-0.55%, while after the optimization was done, the values were reduced to 0.084+/-0.069 and 0.58%+/-0.48%, hence the immense robustness, strong accuracy, and higher fault-tolerance, as claimed by the authors themselves. Their methodology and perspective were quite intriguing with the use of BP neural networks for online price prediction and further model optimizing, which eliminated inflexibility and lack of learning.

Author Listiani [14] has used SVM to predict leased car prices and showed a better accuracy rate than multivariate or a simple multiple linear regression when a large dataset with more dimensions is present, which also reduces overfitting and underfitting. The downside of it was the absence of indicators as mean, standard deviation, and variance to show the changes between multiple linear regression and SVM regression.

Assessments discussed [5]-[14] only analyzed performance statistics of prediction for several acclaimed models individually.

However, Gegic et al. [15] divergently conducted their study by comparing the algorithms Random Forest, ANN, and SVM and finding out the most reliable and accurate predict model, with regard to the 797 records preprocessed car dataset, initially obtained from autopijaca.ba web portal during the winter season in Bosnia and Herzegovina, using a PHP web scraper. Authors, however, made an ensemble modal out of the three algorithms and proved the fact that rather than opting for a single algorithm from the above stated three, which each of them individually had less than 50% accuracy, a hybrid version employs much better potential in predicting the price with their combined accuracy on test data as 87.38%. Although having more samples than 797 could have improved prediction power and 90:10 split probably lead to model overfitting that reduced generalization levels, their articulation of the ensemble model is outstanding and encouraged this research to audition the same using supervised learning regression algorithms.

1.3 Research Gap

Even though it is perspicuous that some researchers have attempted to implement an outstanding model that is capable of predicting prices of used or secondhand cars with decent accuracy levels, still much more undiscovered innovative methodologies are yet to be realized. Most of the research that was conducted before has used only a small number of records as their datasets. For example, Pudaruth [11] considered only 97 records to train and analyze the models, while Noor and Jan [6], Peerun et al. [7], Enis Gegic et al. [8] used 1699, 200, and 797 respectively, which is extremely below than the amount that this research component would be working on, which is about 450,000 records (after preprocessing will be reduced). Hence the issue of models being biased to training data will be reduced since the range of records is higher and will additionally improve the prediction accuracy as well.

On the other hand, most research [5]-[15] have considered only a lower range of car characteristics for the prediction. Through this component, further analysis using specific techniques like feature engineering/feature selection will be done to identify the most relevant attributes that make an influence to the price, even at the slightest level, and consider those features to be as inputs to the considered models. When considering the available research papers [5]-[15], it was vividly noticeable that the authors have only utilized and scrutinized six [9] or fewer classification/regression algorithms when opting for an effective prediction model implementation. At times, authors tend to testify a single type of algorithms competency, let alone even more. They have considered a few algorithms and continue on their focus scoped only towards those algorithm statistics and declare that one or the other is the most ideal in price valuation. Hence the advantage and precedence of multiple algorithm performances were not explored rigorously. In consequence, this research made its aim to inspect 15 variegated regressors to ascertain their price prediction efficiency and proficiency by making necessary refinements for their parameters/hyperparameters during model training and testing phases in order to obtain further success in their executions. The above utterance conclusively depicts the research gap being addressed and rectified. In addition, when analyzing the

research papers further [5]-[14], it is understandable that those authors have analyzed and examined the performance, accuracy, and mean error levels, for only a single ML algorithm, in the context of car price prediction. Even though multiple algorithms were under consideration for those studies, the final deciding model was built based upon the most optimal algorithm. RF, ANN, SVM, multiple linear regression, and KNN were the most popular singular executing algorithms chosen for the task by many. However, Gegic et al. [8] have proved that rather than applying a single model, a composition of two or more algorithms works better in predicting the car prices more accurately with stronger precision. The authors of that research have manipulated three algorithms individually and came to the settlement that the error rates were high, and accuracy levels were less than 50%, although when used as an ensemble model, the accuracy was risen to above 92% and addressed the overfitting problems of certain individual models and improves generalization. Therefore, this research component is engrossed in implementing a hybrid model with four predictive algorithms that have never been attempted as a combination in previously fulfilled explorations.

Exiting research [5]-[15] have failed to uphold their stature when contemplating on developed ML model's API (Application Programming Interface) integration. For a particular model to be successful in its functionality to the fullest, it must be integrated with some sort of a platform that is accessible to business procedures involving end consumers, who are the essential benefiters of these ML model capabilities. API endpoints should effortlessly import and export ML model dependencies if working smoothly is required. Hence, ML models must be incorporated with simple yet effective deployment procedures that are cross-technology compatible. The API implementation must be impeccably and professionally constructed to work with a hybrid ML model as well, which could be more arduous. Thus, this research targeted to produce the triumphant models that would be working as an ensemble, with a practical tendency to be applied upon any chosen technology-infused API so that their service would be realized through a business application UI. Not only the absence of an integrated API but also the presence of a UI for the final evaluated price visualization not being produced should be thrown into the bargain when observing current research [5]-[14]. In [15], they have managed to integrated the final model with

a Java-based UI, although that itself is not related to any web/desktop application that can be used by consumers and does not have a higher ultimatum such as e-commerce process encouragement. Table 1.1 below depicts a simple, summarized comparison between most similar research and the developed system.

Table 1.1 Comparison of key previous researches

Research	**Complex dataset having a higher number of records**	**Consideration of more features affecting the price of a car**	**Comparison of a higher number of (>4) algorithm models (This research – 15)**	**Implementation of an ensemble model having a composition of various models**	**API endpoint amalgamation and front-end UI visualization**
Research A [7]	✗	✗	✗	✗	✗
Research B [8]	✓	✓	✗	✗	✗
Research C [9]	✗	✗	✓ (6)	✗	✗
Research D [10]	✓	✗	✗	✗	✗
Research E [11]	✗	✗	✗	✗	✗
Research F [13]	✗	✗	✗	✗	✗
Research G [15]	✗	✗	✗	✓	✓ (UI only)
Tievs Price Prediction	✓	✓	✓	✓	✓

1.4 Research Problem

It is apparent that the second-hand or used car population has grown higher than the brand new car population having the causes as people's fund management issues and popularity of leasing approaches. Thus, it is a delicate task to decide on an affirmative price that is fair both to the car reseller and the buyer. It is further tedious when the market price hunting needs to be done by undergoing it manually, by the reseller themselves, through extensive and vast researches, while wasting precious time out of their schedules. It is undoubtedly a strenuous and irritative matter that yearns for a simple resolution which, in turn, is positively realizable.

Even in online classified platforms, customers still find it difficult to set an appropriate price value for their car classified advertisement. As a result, customers occasionally tend to state prices as $0, which is extremely unacceptable, especially if the classified advertising platforms requires a quality reputation. The Fig 1.5 and 1.6 below displays how customers have clarified $0 as their price values for the reselling second-hand cars, within the majorly famous platform 'Craiglist'. This is unwaveringly erroneous behavior.

Figure 1.5 Unrealistic car classified price part 1

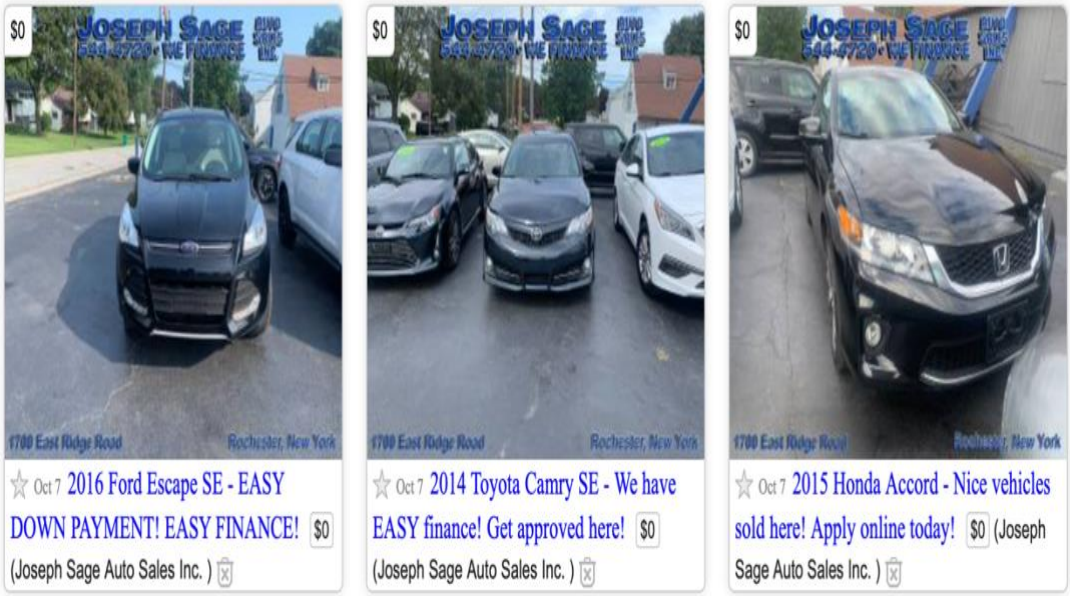


Figure 1.6 Unrealistic car classified price part 2

That is why it is important for a price evaluation facility to be integrated within the portal's ad submission venue for customer requirement catering. When analyzing the existing major classified platforms around the world, most of them did not incorporate any sort of price prediction mechanisms within them hence making the concept fresh altogether.

Regardless, it was noticeable that the ones having some kind of a price evaluation process integrated into their advertisement submitting the form depended on a third-party website crawler that extracts price values from a different platform, which is out of their classified system's control. That method seemed sufficient and practical at first, but when examining it deeper, a contrasting understanding emerged. If those third-party APIs and servers were to fail or face any interruptions, their service would not be accessible as a whole.

Moreover, if our classified ad platform decides to expand or contract the car attributes being considered in the advertisement form (and for price prediction), that would not be possible due to the rigid attribute types accustomed by the web price crawler. Failing requests from heavy network traffic and future system augmentation

immobilization are some more possible issues that could be faced. Hence an advertising platform providing its own independent product price prediction mechanism is evidently more convenient and currently obscure.

Besides, by utilizing such a prediction mechanism built in-house, the system developers would gain authority and control over application risk mitigation and other procedures as well. Thus, the development of ML models to take up the challenge is indeed necessary, although not thoroughly assessed yet in all Information Technology (IT) domains.

Scientific experimenters have attempted researches to build such ML models that have the correct intelligence and performance for price prediction of used cars in the past few years [5]-[15]. When analyzing through those studies, many ML models designed either from regression or classification procedures have been sophisticatedly investigated, and the results produced have been successful in some scenarios, although others were not commendably appreciative, as discussed in the previous section 'Background Literature Survey'.

Most of them have not considered complex datasets with a large set of records for training and testing of the models, and even though, unquestionably, more data will be making the model highly accurate. Analyzation of features that contribute to the price value estimation needs to be extended from four or six features, all the while maintaining simplicity of the model, which was not thoroughly investigated by most researchers, except for [9], in fear of prediction model complexity. Similar to accuracy, speed is a prominent factor in prediction as well, although it hasn't been explored regarding car price prediction in previous experiments, which is a perceptible gap in research.

It is distinguishable when observing the methodologies of [5]-[15] that they have examined either a single algorithm or several (between 2 to 6) algorithms for their prediction capability. Sophisticated analyses contemplating a higher number of models have not been prompted throughout their research, maybe due to the scope or

computational resource issues. However, if the most suitable algorithm is to be recognized, the range of the comparative studies should be amplified, although it has not been a reality yet. Various hyperparameters relating to different algorithm models can be familiarized, and how their changes may affect each model's accurate performance can be identified as well.

Proven by authors of [15], using a constituent single learning model for prediction realizes less accuracy and more error gaps than using a composition of predictive models, which gives superior forecasting power in comparison. Regardless, a majority of the experimentations have appraised models individually by inspecting if the desired accuracy levels are being depicted or not, according to dataset deviations and model parameter tweaking.

For example, researchers such as [5,8,9,11,12] have taken into account different ML algorithms when conducting their research studies and never did opt to apply for a hybrid model and analyze the results to see if the performance increases or not. They compared the performances of the models separately by training them with the same dataset and finally concluded the model having the best accuracy and least error rates as the best price prediction model to be used in the market.

Nonetheless, if all of the best models were to be combined together as a single ensemble model to harness all of their strengths to provide the prediction efficacy, their accuracy levels would definitely increase. For example, when two or more people are involved in decision-making, an exceptional solution can be acquired quickly, although it is the opposite when there is only a single person trying to come up with solutions to a problem.

The same theory adds up to ML models as well. Thus, utilizing ensemble models is obligatory if precision is valuable, though the usage is extremely rare in the field of car price prediction, and the combination of the appropriate models to be considered is plenty for exploration. The following diagram from Fig. 1.7 elicits the main types found in ensemble learning.

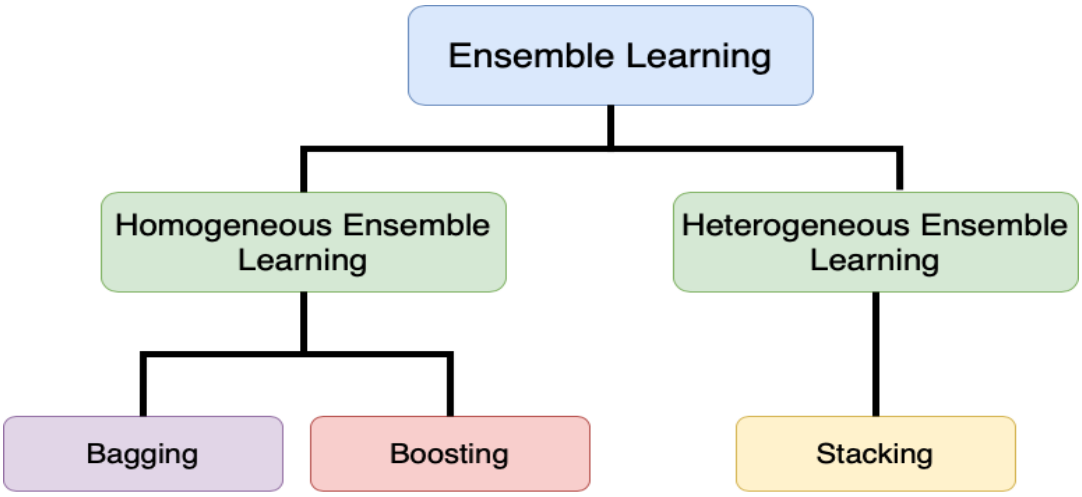


Figure 1.7 Main types of ensemble learning

Another problem with the past studies done is that they only intended on developing the relevant ML model having a higher efficiency level in the prediction of a price value. Their experimentations ended from there onwards. Integration with a front-end application, let alone with an API to facilitate model amalgamation and prediction result providence for visualization purposes through HTTP methods and endpoints, was extremely scarce and not endeavored further.

Moreover, model and API cloud deployment was overlooked as well. However, if there is no way a consumer service for customers could be bestowed by a price prediction model's prosecution, the model itself is futile and useless. All excellent or moderate research [5]-[15] have attained models with the best accuracies and lowest error rates but have not incorporated any API's or client applications for model deserialization and collaborative execution.

1.5 Research Objectives

Main objective

- The ulterior objective of this research was to implement a dynamic price predicting system, within the scope of one year, using a hybrid predictive ML model, to recommend a selling price value for specific products (Cars) being advertised on the 'Tievs' classified advertising platform.

When diving into the bigger picture, upon the accomplishment of this objective, it will serve with high importance as one of the major fresh inclusions for the overall 'Tievs' SMART online classified advertising system, enabling it to surpass the existing applications that publish classifieds. Most of the available systems do not include an internal ML model-driven process to predict a price of a product and suggest the customers with it through the application's user interface (UI). Hence it has become the main objective of this dissertation-based research to facilitate 'Tievs' with that enhancement so that it astoundingly emerges from other competitors. However, due to scope and time constraints inflicted, the system will implement these functionalities, targeting one type of classification, which is Cars. Subsequently, this component will be focused on predicting the reselling price of the used or second-hand car and displaying it as a suggestion, according to the specifications depicted by the customers on the advertisement form they fill prior to submission. Undoubtfully, the overall population of cars throughout the world has increased when compared to earlier years and still continues to do so [1]-[4]. Hence the reselling rate has become towering due to fewer funds people have at their disposal and the urge to opt for a leasing contract between themselves and the seller. In that situation, after the completion of the whole value payment, the buyer is capable of reselling the vehicle themselves. In these circumstances, people who intend to sell their vehicles and the ones who seek to buy could be swindled by fraudsters and confront disregard for their money due to unreasonable prices tagged. Ostensibly, the significance of an accurate, reliable, and efficient price prediction system for reselling cars is portraited, thus the main objective of this proposed research study.

Specific objectives

In contemplation of achieving the main objective, there are certain sub-objectives that need to be addressed fundamentally.

1. A rigorous analysis of the used car attributes through data engineering and exploratory data analysis to identify the significant, influential factors of the dependent variable price and transformation of those values according to model requirements.

2. Experimentation and a comparative study on 15 different regression algorithm models to compare their accuracy and error rates and verify each of their price prediction power by thorough training and testing.

3. Optimum tuning of hyperparameters values of all the regressor models being considered using various quality mechanisms.

4. Reducing overfitting and underfitting and increasing model generalization in order to predict unseen or future observations.

5. Analyzation of prediction accuracy and error criterion to confirm the top four model strengths, and if not satisfactory, re-iterating the training and testing executions.

6. Implementation and investigation of an ensemble model with a novel combination of triumphantly performing regressor models to predict the price of a car more accurately and argue about its usability and reliability.

7. Conclusive and quality implementation of a robust API to manage the HTTPS request calls targeted towards the final integrated ensemble prediction model.

8. Cloud environment deployment of the model and API with relevant dependencies and development of the front-end application UI components with its associated backend services to facilitate prediction model integration.

2 METHODOLOGY

2.1 Methodology

2.1.1 Theoretical research methodology

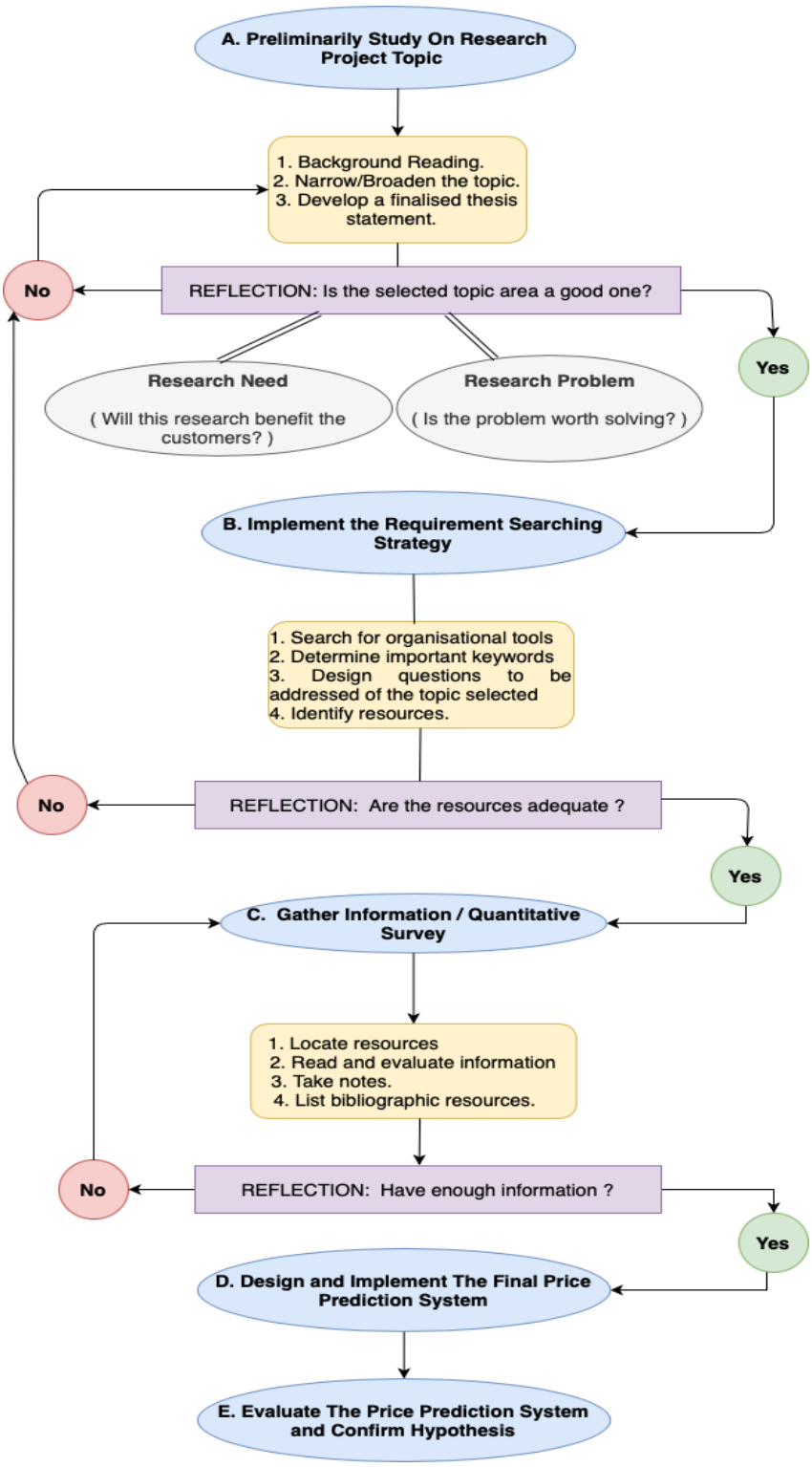


Figure 2.1 Research methodology diagram

The theoretical methodology followed from the start of this research to the end is shown by the above displayed Fig. 2.1. As the first step of the process, a preliminarily study was conducted on the project's main research topic being 'Tievs: Classified advertising ameliorated using machine learning techniques" to identify the existing research done, the topic feasibility, scope, any problem arising or already occurred from those research areas, and any visible research gaps.

This thesis was based on one of the main components surrounding the whole classified advertising application, which is the implementation of an application-correlated dynamic price prediction system. Determining the specific component's topic area's appropriacy was done through a reflection phase. If not satisfied, again, the processes such as further background reading, broadening or narrowing of topic scope, and finalization of the thesis statement were done.

Simultaneously, the two concepts, which are inferring the research's need towards the end-users and analyzing the prominence of the research problem and benefits gained by solving it was contemplated. Afterward, the requirement searching and implementing strategy was undertaken to search for innovative and updated algorithms, tools, and technologies that can achieve a solution for the determined problem.

Similarly, resources that could be an advantage for the research procedures were identified, and important keywords were examined as well. Then, a rational realization was conducted to decide if the resources acquired were sufficient and if not, the above process was iterated again. Then the data gathering section came into place. In this phase, not only the ML model required dataset obtaining was conducted, but also other literature information and bibliographies were evaluated, and notes were taken for ensuing processes. Specific Surveys or questionnaires were not required for the initial startup of the solution implementation; hence such weren't created or distributed. Only a consumer feedback survey was to be distributed by the end of the whole integrated system fulfillment. Below mentioned 'Dataset Acquiring' section emphasizes the utilized dataset for ML processes.

Dataset acquiring

Since manual data collection on used cars would be tedious and time-consuming [5]-[10], this research component utilized an openly available online Kaggle dataset. The dataset was initially published and updated by Austin Reese, which is scraped from one of the popularly listed online classified advertisement platforms, Craiglist, which is sparsely known and being used by many customers within the United States.

The 'Used Cars Dataset' (vehicles.csv) conforms to about 450,000 records (441,396 to be exact) about sold cars of 43 various brands through the Craiglist website and includes 26 attributes that define distinctive characteristics of the second-hand cars, that of which some were excluded due to irrelevancy or empty fields. The dataset is being updated within every three-month interval hence the appropriacy for the implementation of the price prediction model since the most recent price values of second-hand vehicles must be attained in order to deliver an accurate and precise result for the customers requesting the service.

The reason that this particular dataset is ideal for the implementation of the car price prediction component of the Tievs application is that the application is fundamentally being built by targeting the consumers of the USA. Initially, our reference person, who has been a resident of the US for more than 10 years, has requested the requirement of that which a more ameliorated classified advertising portal having novel and sophisticated IT technologies that could be implemented to cater to the customer necessities adequately without any inconveniences.

Therefore, our research project team had decided to build such a system targeting their market in order to gain some business-wise profits by addressing the prevailing issues faced by the consumers through their native classified advertising portals. Thus, deploying the Tievs application incorporating essential components into production is indisputably the ultimate goal. Hence, the information about the USA sold cars contained in the utilized dataset itself is appropriate for the progress of this matter in hand.

Recommencement of theoretical research methodology

Concluding the necessary information-gathering phase, the next step is to speculate about the validity and sufficiency of the already acquired knowledge and information. If it is not adequate, the previously conducted phase will again be looped, having to locate more new resources, read and evaluate more new information, list bibliographical resources identified, and taking more descriptive notes. However, if the recorded information is more than satisfactory, the next step will be prompted immediately. Designing and implementing the ultimate and final price prediction system for the Tievs application is a challenging and sophisticated process all in and out, which most conceivably starts from the dataset gathered for the training and testing purposes of the considered ML models must be thoroughly cleansed in order to startup with the model implementation, those being either classifiers or regressors. The accuracy levels and error rates must be carefully dealt with in order to maintain the quality of the price prediction system. After the ML model implementation procedures, the API server creation step occurs, and that too must be developed meticulously utilizing the most convenient programming framework after a complete investigative venture on that.

After all backend procedures have been accomplished, the next would be to move in on the client-side application development. Before the development starts, existing technologies must be clearly analyzed and examined for their ability to provide an efficient and effective ground structure for the prospective classified advertising system, which actually should be much enhanced than the prevailing systems in the world wide web (WWW). The most optimal technology or the programming language must be utilized for precise and rapid development since the scope of the research is itself limited. Following the successful implementation of the price prediction system, it is then the time for the initial hypothesis analysis to determine whether the raised problem has been solved using the assumed methodology, concepts, and techniques. If the hypothesis has been proven true, it can be addressed as affluent research. Else, the results can further be examined to find what went wrong and why, and recommendations of reconciling mechanisms can be conferred as well.

2.1.2 High-level system design

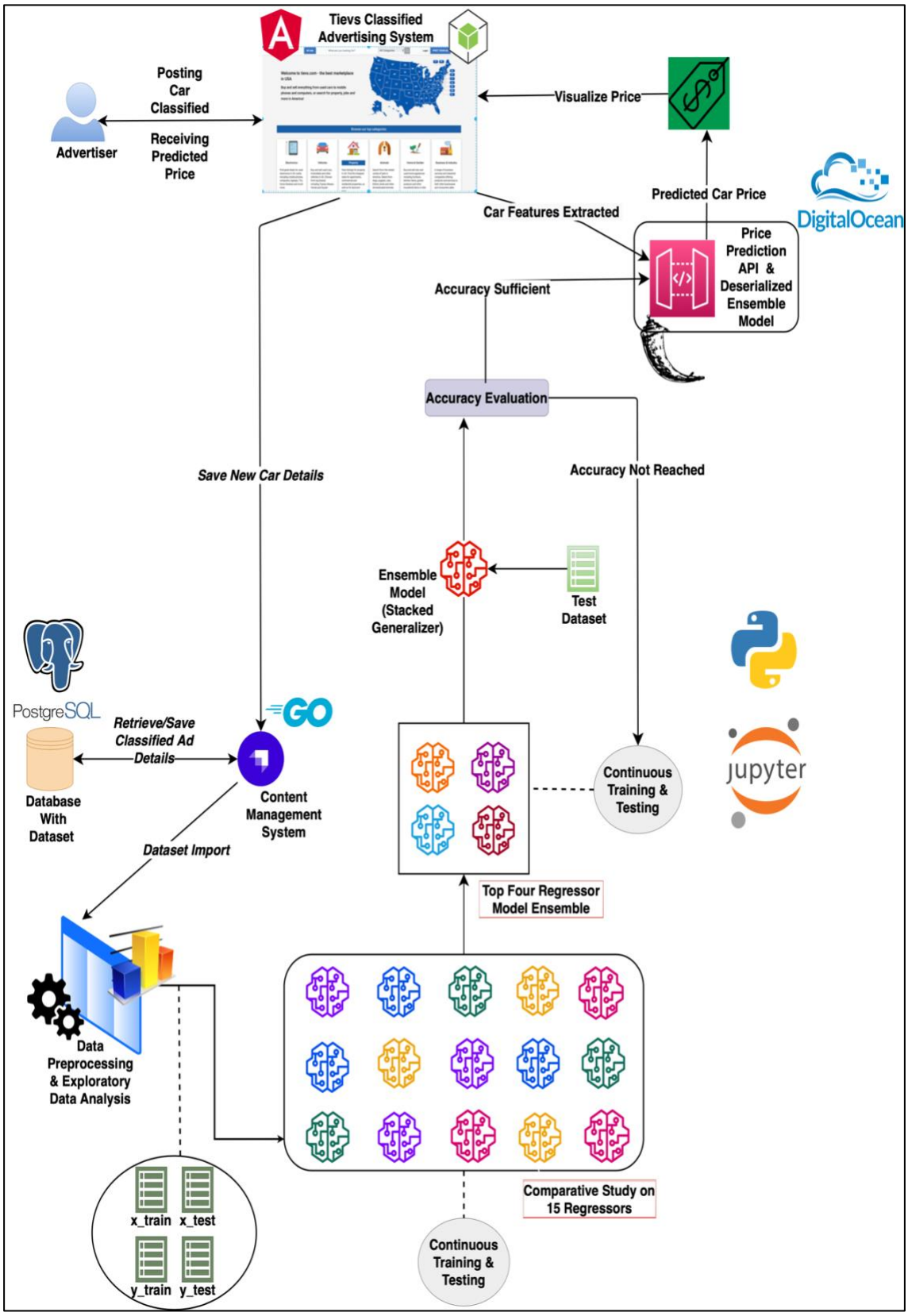


Figure 2.2 Tievs price prediction system overview diagram

The above-shown Fig. 2.2 depicts the overall high-level system diagram of the proposed price prediction system that was built in order to provide a price valuation regarding the used cars that are being published in Tievs as car classifieds. Python was used as the main programming language with Jupyter Notebook for the implementation throughout. The following explanation will accentuate the high-level system diagram contrived steps incisively.

1. First, the acquired dataset was imported from the PostgreSQL database into the Jupyter Notebook environment, and it underwent exploratory data analysis parallel to various attribute selection techniques, feature engineering techniques, and other data preprocessing methods to understand and cleanse the dataset prior to transferring it into the models.

2. Then, it was split into two major subsets as X and Y, according to the independent variables and the dependent variable (price). Thereafter, it was again split into more sets such as x_train, x_test, y_train, y_test at a 75:25 split ratio for training and testing. K-fold cross-validation was additionally utilized to assess model effectiveness with regard to overfitting and hyperparameter tuning.

3. A comparative study was conducted targeting 15 different regressor models to perceive their performance with regard to price prediction while utilizing a complex dataset. Those models were trained and tested using the same train and test datasets continuously until satisfactory results appeared.

4. The optimum model hyperparameters were identified and adjusted accordingly to the dataset characteristics using specific methods. The training processes and, likewise, the testing processes were applied repetitively for each of the models for better learning perception.

5. The train and test dataset's accuracy levels and error rates were analyzed for individual models, and afterward, the top four victoriously performing regressor models were identified.

6. The top four models were amalgamated together to form an ensemble method, which resulted in a hybrid regression model. Using the test dataset, the accuracy of the hybrid model was analyzed, and the surfaced refinements that were necessary to be completed were done, and as a result, the process navigated back to the training of the four exceptional models once more.

7. Simultaneously, an API was built using Flask python micro-framework in order to facilitate the communication between the model execution and result incorporation, along with a client-side application. The API endpoint implemented was a POST since in order for the prediction to happen, first of all, from the client-side, the forecasting car characteristics must be obtained and passed through to the API.

8. Subsequently, succeeding the precision of the hybrid model, it was serialized using the pickle python library and again deserialized inside the API so that it can be integrated with the Flask ML API in order to provide the prediction results for the front-end application. The API containing the model was then deployed in the DigitalOcean cloud environment for admiring scalability and 24/7 availability.

9. Consequently, a cloud-deployed Tievs front-end application was built using the Angular framework. Through the implemented classified advertisement submission form UI, the essential prediction features could be extracted and fastidiously analyzed to facilitate the API endpoint invocation and model price prediction delivery.

10. After the ensembled stacked regressor has given the response through the API, it was able to visualize the final price range repercussion to the customer, within the advertisement form, near the price input section. New car classifieds being submitted into the application, having car attributes and other details, will be saved in the database and will be utilized for the purpose of retraining the prediction model.

2.1.3 Software development life cycle

This research was guided throughout the year, with the infamous continuous delivery and continuous development aspirated, change welcoming, methodology approach, Agile. Continuous delivery, face-to-face interactions, motivated personnel, change embracement, working software deliverables are some of the main considerations in the agile procedures and those of which were heeded within the development of this proposed price prediction system. Although a set plan was constructed to aid for the time management and delivery schedule alignments, impediments did occur, and the necessary adaptations were taken place in a timely manner, without interrupting the continuous development processes. When conducting this research, grey areas, with many uncertainties either in the methodology or in the techniques, were encountered and responded effectively and positively. Due to agile processes, it was easier to contemplate the obstacles by making them the opportunities for success itself. In conclusion, Scrum, which is a lightweight version of the vast Agile framework, was be utilized to cater the incremental deliverables with high quality during short periods along with overall team and project management.

2.1.4 Technologies and tools utilized

Tools

- PyCharm
- Jupyter Notebook
- WebStorm
- VSCode
- Brackets
- Postman
- DigitalOcean
- Cloudflare
- Supervisorctl
- Loadmill

Technologies

- Python
- Scikit Learn ML regressors
- Flask quart framework
- HTTP/2 (HTTPS)
- Seaborn, NumPy, Matplot, Pyplot
- Ensemble learning
- Angular framework
- Typescript
- NodeJS
- PostgreSQL

2.1.5 Project requirements

Functional requirements

- Ensemble model implemented after 15 regressors comparative study should provide precise prediction accuracy having low error rates with relevance to car features considered.
- Price prediction ML API and ensemble model integrated without prompting errors.
- Successful cloud deployment of both API and model without failures.
- Convenient implementation of the front-end application features.
- Successful customer navigation to the advertisement posting section.
- Correct visualization of the advertisement form components for a convenient depiction of the various characteristics of a car.
- Accurate functionality of the 'Predict Price' button.
- Extraction of the car characteristics into the prediction model.
- Precise price prediction according to extracted details.
- Shall clearly present the predicted price to the customer.

Non-functional requirements

- Accuracy
- Less response time
- Scalability
- Efficiency
- Effectiveness
- Ease of use
- Time-saving
- Adaptability
- Accessibility
- Maintainability

2.2 Commercialization Aspect of the Product

As Fig. 2.3 shows, the incorporation of a price prediction system within an online car classified portal can lead to higher and faster sales of the cars that are being advertised since the tagged price values are much more convenient, reliable, and realistic when comparing with other similar cars within the market and the buyers are more obliged to make a complete purchase consequently. Hence the business revenue of the application as a whole will be increased as well due to the higher number of customer visits and utilization. Customers would not need to do extensive research when clarifying a price value by exiting the application, which results in a decrease in the bounce rate. They would simply receive an approximated price value at the application venue itself and will be thoroughly satisfied with the service that saved their valuable time out of their busy schedules. Moreover, the inclusion of such a price prediction system within the application will give it a towering competitive advantage among other parallel systems that provide equivalent efforts in satisfying customer necessities. An advertising platform providing its own independent item price prediction mechanism is more convenient from an application developer's point of view due to intrinsic control authority over the price valuation, which is currently obscure since most similar platforms use website crawlers and face unnecessary third party dependencies, heavy network traffic, and future enhancement immobilization.

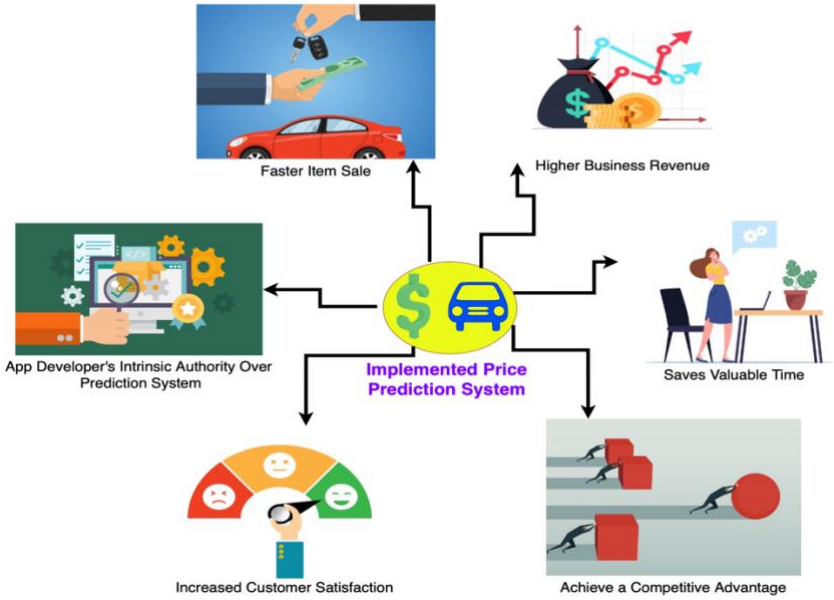


Figure 2.3 Commercialization aspect of the product

2.3 Implementation

2.3.1 Dataset realization

Storage of the dataset will be done by using the PostgreSQL database environment due to its winning aura against other infamous database systems such as MySQL, MongoDB, Oracle DB, MariaDB, SQLite, Firebase, etc. PostgreSQL tends to be faster in reading and write speeds regarding data and information, occupies less space as storage, utilizes memory capacity to the lowest, and minimum CPU power consummation is also a notable characteristic for it radiates among others. Hence, it was chosen for this research's path towards success within the data storage aspect. The dataset view was first and foremostly analyzed using the Jupyter Notebook server UI platform, and below Fig. 2.4 depicts the first five rows and columns of the selected dataset, which is visualized through python pandas library commands.

```
In [6]: dataset.head()
Out[6]:
```

price	year	manufacturer	model	condition	cylinders	fuel	odometer	title_status	transmission	VIN	drive	size	type	paint_color
9500	2003.0	chrysler	town & country	excellent	6 cylinders	gas	30376.0	clean	automatic	NaN	fwd	mid-size	mini-van	blue
28990	2018.0	gmc	acadia sle-2 sport utility	good	6 cylinders	gas	20736.0	clean	other	1GKKNLLS2JZ178034	fwd	NaN	other	white
16590	2013.0	lincoln	mkz sedan 4d	good	NaN	gas	61087.0	clean	other	3LN6L2G99DR819673	fwd	NaN	sedan	red
28990	2017.0	gmc	acadia slt-1 sport utility	good	6 cylinders	gas	18041.0	clean	other	1GKKNMLS2HZ305977	fwd	NaN	other	NaN
20590	2017.0	lincoln	mkz premiere sedan 4d	good	NaN	gas	36436.0	clean	other	3LN6L5A97HR666209	fwd	NaN	sedan	NaN

Figure 2.4 Dataset first five rows

The dataset being constantly updated is important essentiality if it's to be assimilated into an ML price prediction model. That is because the customers who are acquiring the services of the prediction model should always receive the current market roaming price trends with respect to the particular car model they are in the process of selling by inserting the classified. Market trends can be highly influential towards the reselling price of a used car. This condition is satisfied by the utilized 'Used Cars Dataset' since it is repeatedly being updated on a six-month basis through the Kaggle portal by its

initial creator. Beneath stated link directs to the webpage where the dataset is presently located in Kaggle.com, and Fig. 2.5 illustrates the introductory view of it.

Dataset Web Link - https://www.kaggle.com/austinreese/craigslist-carstrucks-data

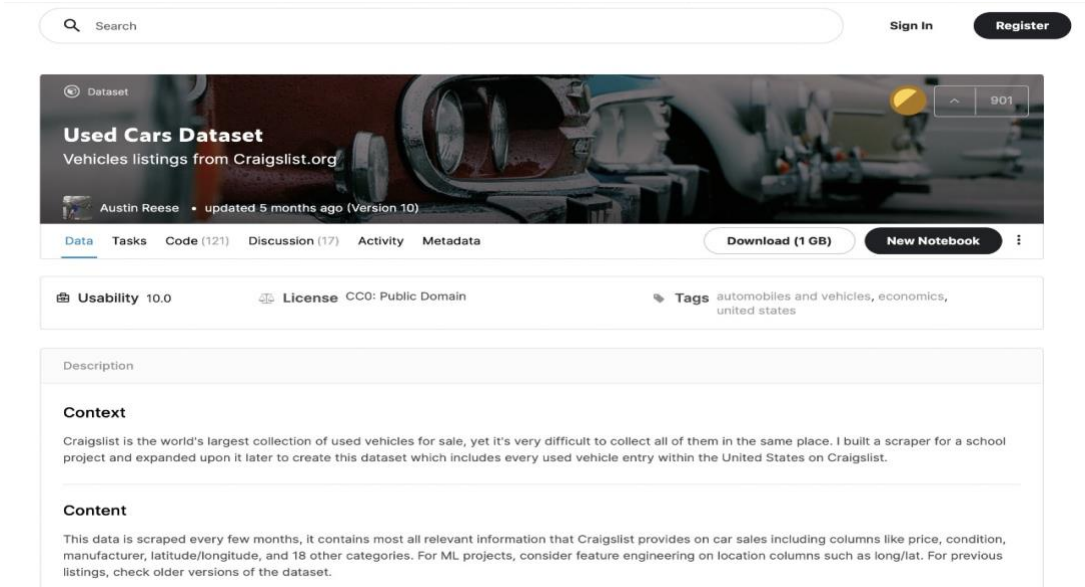


Figure 2.5 Dataset Kaggle view

Below Fig. 2.6 screenshots describes the columns that are present within the utilized dataset with their respective column types as well. Only the relevant columns will be considered for model prediction processes after thorough data preprocessing.

```
In [52]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 441396 entries, 0 to 441395
Data columns (total 25 columns):
id              441396 non-null int64
url             441396 non-null object
region          441396 non-null object
region_url      441396 non-null object
price           441396 non-null int64
year            440359 non-null float64
manufacturer    423019 non-null object
model           436057 non-null object
condition       257554 non-null object
cylinders       253231 non-null object
fuel            438515 non-null object
odometer        437018 non-null float64
title_status    432451 non-null object
transmission    438769 non-null object
VIN             277315 non-null object
drive           307747 non-null object
size            125812 non-null object
type            346047 non-null object
paint_color     308374 non-null object
image_url       441343 non-null object
description     441340 non-null object
county          0 non-null float64
state           441396 non-null object
lat             434669 non-null float64
long            434669 non-null float64
dtypes: float64(5), int64(2), object(18)
memory usage: 84.2+ MB
```

Figure 2.6 Dataset columns

Table 2.1 Descriptive statistics of categorical variables

Features	Count	Unique Count	Top	Top Frequency
url	441396	441396	All unique	1
region	441396	403	columbus	3742
region_url	441396	412	https://bend.craigslist.org	2992
manufacturer	423019	42	ford	70467
model	436057	30082	f-150	8006
condition	257554	6	good	122203
cylinders	253231	8	6 cylinders	94528
fuel	438515	5	gas	367217
title_status	432451	6	clean	418054
transmission	438769	3	automatic	357948
VIN	277315	123662	KMTG34LE6LU049741	232
drive	307747	3	4wd	133843
size	125812	4	full-size	65569
type	346047	13	sedan	94094
paint_color	308374	12	white	81895
image_url	441343	249902	url	4990
description	441340	372923	All unique	1
state	441396	51	ca (California)	52089

Table 2.2 Descriptive statistics of numerical variables

Features	Count	Mean	Std.	Min	Max
price	441396	6.40e+04	1.11e+07	0.00e+00	3.74e+09
year	440359	2011.49	9.27	1900.00	2022.00
odometer	437018	2.25e+05	1.60e+07	0.00e+00	2.00e+09
county	0	nan	nan	nan	nan
lat	434669	38.47	5.86	-73.08	84.62
long	434669	-94.57	18.24	-175.32	173.89

The above-depicted Table. 2.1 and Table. 2.2 shows some of the descriptive statistics of the categorical and numerical attributes containing in the raw used car dataset prior to any processing that is applied.

2.3.2 Data preprocessing & exploratory data analysis

This research conducted both data preprocessing and exploratory data analysis together since both are quite interrelated and both depend on the other's results when tweaking the dataset as necessary. In order to get some insight on how the dataset can be utilized along with the models to be analyzed, visually descriptive methods can be implied. It enabled this research component to better identify the correlations between the existing car characteristics with the relevance to their prices. Basically, a viewpoint

can be established on what ranges the attributes in the dataset reside, the diversity of each and every feature in the dataset, and to determine the correlation between each feature considered, including the target feature price itself, significantly vary or stay dense. For the purpose of visualization, bar charts, box plots, scatter plots, distribution tables, etc., were manipulated. By conducting this exploratory data analysis process, it helped filter out unnecessary columns, unrealistic data ranges, and outliers that may cause misinterpretations within the prospective ML models. Data preprocessing can be known as the most challenging and major part of the whole experimentation process [15] since the increase in performance of a prediction model would be the sufficient amount of data preprocessing done exceptionally as to transform raw data into a more clear, understandable, efficient and useful format through outlier removal and data standardization. These processes were fundamentally done using Jupyter Notebook and the necessary python libraries that needed to be imported, such as Pandas, NumPy, Seaborn, Matplotlib.

- First of all, filtering out the irrelevant columns that do not affect the price of a car was done using the python commands shown below in Fig. 2.7. (Ex: URL, region, region_url, county, image_url, lat, long, VIN, description, state, and id)

```
In [57]: dataset.drop('county',
           axis='columns', inplace=True)

In [58]: dataset.drop('image_url',
           axis='columns', inplace=True)

         dataset.drop('url',
           axis='columns', inplace=True)

         dataset.drop('region_url',
           axis='columns', inplace=True)

         dataset.drop('lat',
           axis='columns', inplace=True)

         dataset.drop('long',
           axis='columns', inplace=True)

In [59]: dataset.drop('VIN',
           axis='columns', inplace=True)

         dataset.drop('description',
           axis='columns', inplace=True)

         dataset.drop('region',
           axis='columns', inplace=True)

         dataset.drop('state',
           axis='columns', inplace=True)

         dataset2.drop('id',
           axis='columns', inplace=True)
```

Figure 2.7 Insignificant column removal

- Through this feature engineering or feature selection [16], the retained impactful features that had at least a slight impact on the price were the year, manufacturer, model, condition, cylinders, fuel, odometer, title status, transmission, drive, size, type, paint color (overall 13 final estimators). Following Fig. 2.8 shows the correlation matrix heatmap between the numerical variables before irrelevant columns were removed.

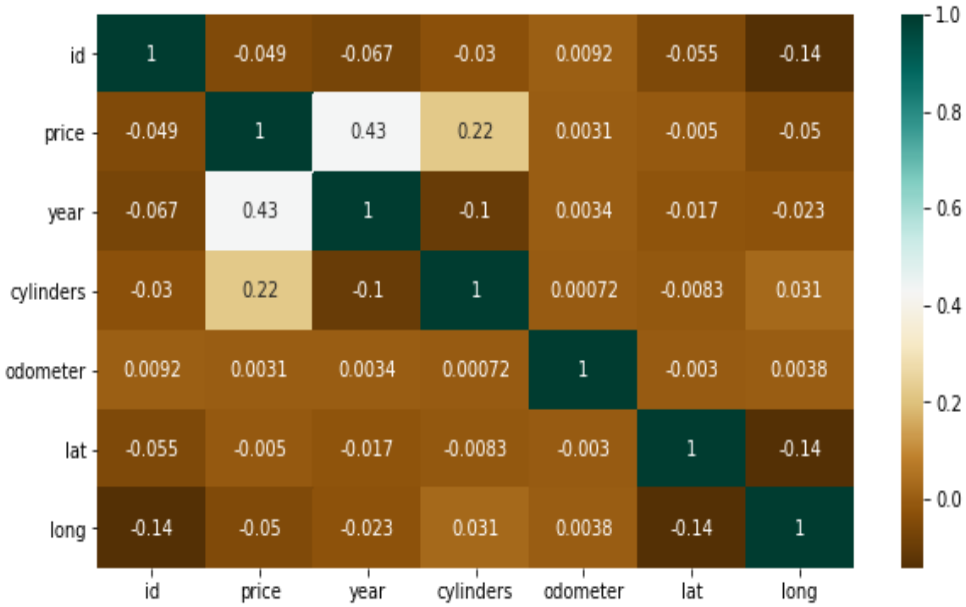


Figure 2.8 Correlation heatmap between numerical variables

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \qquad (1)$$

- The above-stated correlation equation (1) is used especially in predictive statistics to identify highly related attributes, but not at all times [17]. The values range between 1 (positive relationship) to -1 (negative relationship), and 0 suggest no relationship. As the name ‘correlation coefficient’ name implies, the statistical relationship between variables are acquired, and in this research, as it was hypothesized, according to the above Fig. 2.8, year, cylinder number and odometer have more impact on the price from numerical variables with the values respectively being 0.432607, 0.222595, 0.003122.

- Rather than eliminating a whole record just because it has some null values is unnecessary. Therefore, filling attribute null values with respective mean/mode values was done next which can also be called imputation. Here, year and odometer column mean values were calculated as shown by below Fig. 2.9 and then those values were replaced with the null values both year, and odometer columns had since they were numerical variables.

- As for the categorical variables, the mode values were accounted for and categorical column null values were replaced appropriately by the python command shown in Fig. 2.10. The price column did not have any null values to be addressed. Fig. 2.11 displays how it was confirmed that all null values had been dealt with or not.

```
In [60]: dataset["year"].mean()
Out[60]: 2011.4915080650105

In [61]: dataset["odometer"].mean()
Out[61]: 225255.58168313434

In [62]: dataset["year"].fillna(2011, inplace = True)

In [63]: dataset["odometer"].fillna(225255.5, inplace = True)
```

Figure 2.9 Numerical column null value handling

```
In [67]: dataset2 = dataset.apply(lambda x: x.fillna(x.value_counts().index[0]))

         del dataset
```

Figure 2.10 Categorical column null value handling

```
In [68]: dataset2.isnull().sum()

Out[68]: id              0
         price           0
         year            0
         manufacturer    0
         model           0
         condition       0
         cylinders       0
         fuel            0
         odometer        0
         title_status    0
         transmission    0
         drive           0
         size            0
         type            0
         paint_color     0
         dtype: int64
```

Figure 2.11 All null values handled

- How the price has been distributed among all used car records in the dataset was analyzed, and it was decided to remove the records having unrealistic price values. Some were in higher ranged outliers, as illustrated Fig. 2.12 diagram showing the mean prices related to manufacturers, and even 0 prices were present as well.

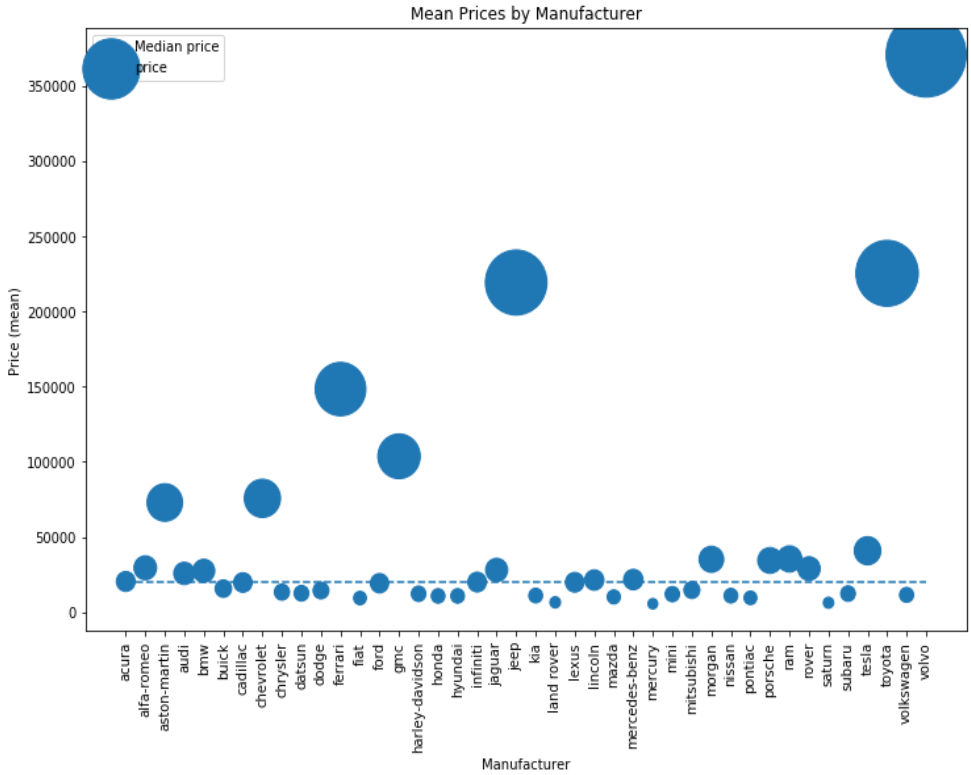


Figure 2.12 Mean prices by manufacturer

- The records where price = 0, price < $750, and price > $100000 were eliminated by the below depicted commands of Fig. 2.13. Then, the dataset price distribution could be visualized by the below shown Fig. 2.14 graph.

```
dataset2 = dataset2[dataset2.price != 0]

dataset2 = dataset2[dataset2.price >= 750]

dataset2 = dataset2[dataset2.price <= 100000]
```

Figure 2.13 Price range normalization

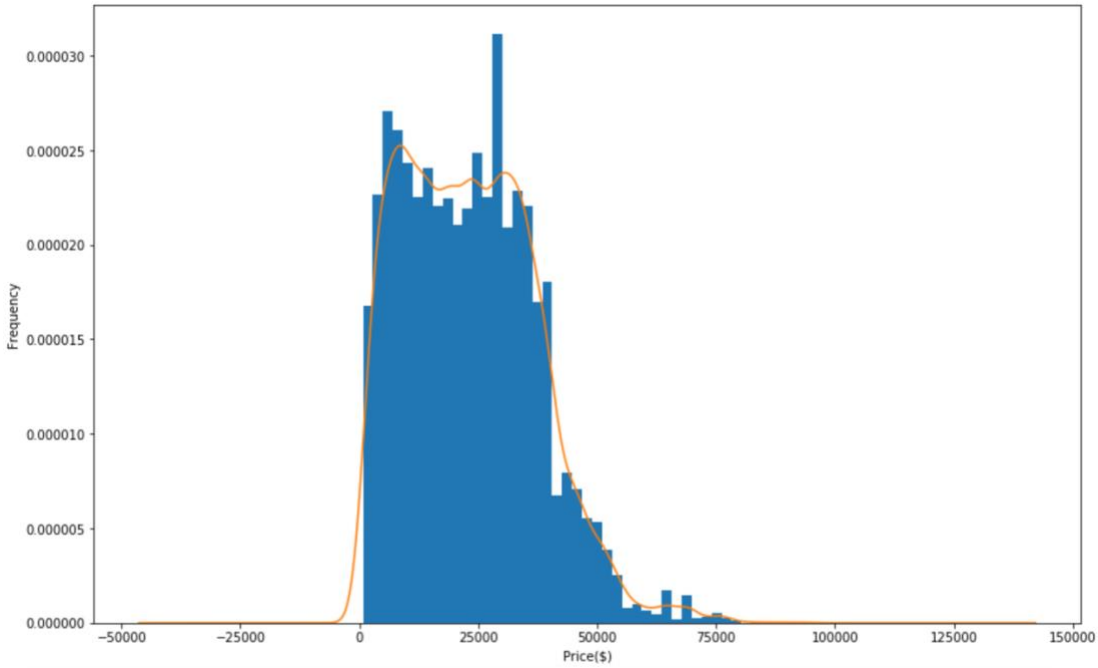


Figure 2.14 New price distribution

- The different manufacturers or brands and their respective counts for the dataset containing car records are illustrated as the below Fig. 2.15 displayed bar chart, and by swift analysis, one can see that Ford, Chevrolet, and Toyota branded cars are the most commonly populated. Hence, the records relating to those manufacturers were considered for further procedures by Fig. 2.16 mentioned commands.

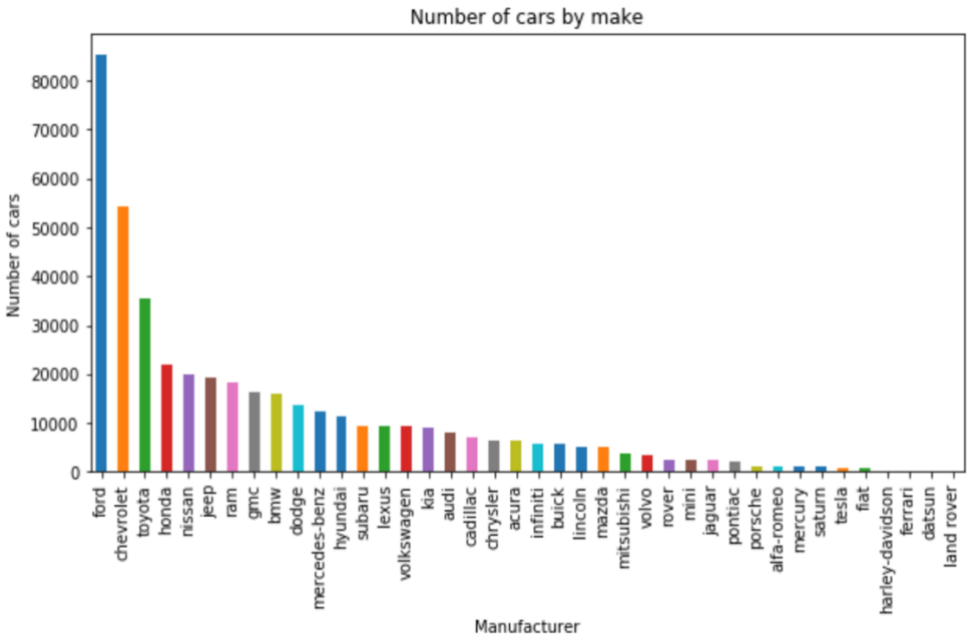


Figure 2.15 Number of cars by manufacturer

```
dataset2 = (dataset2.loc[dataset2['manufacturer'].isin(['ford','toyota','chevrolet'])])
```

Figure 2.16 Selection of records pertaining chosen car manufacturers

- Due to computational resource limitations faced, the targeted dataset scope had to be additionally minimized for 14956 records. The model type being f-150, which was the mode value of that column, was chosen to filter out the records steadily using the python commands in Fig. 2.17.

```
options = ['f-150', 'f150']

dataset2 = dataset2[dataset2.model.str.startswith(tuple(options))]
```

Figure 2.17 Car model filtration

- Next, the year column had to be managed accordingly. It is because the dataset seemed to have year value outliers as in cars made in 1900 and even in 2022, which is extremely unplausible. Hence, for this study, the cars made from 1975 to 2021 seemed ample enough for price prediction model creation and was refined using the following command in Fig. 2.18. After the refinement was done, the years that most cars were made and their frequency within the dataset could be exemplified as in the below bar chart shows by Fig. 2.19.

```
dataset2 = dataset2[dataset2.year >= 1975.0]
dataset2 = dataset2[dataset2.year < 2022.0]
```

Figure 2.18 Year range refinement

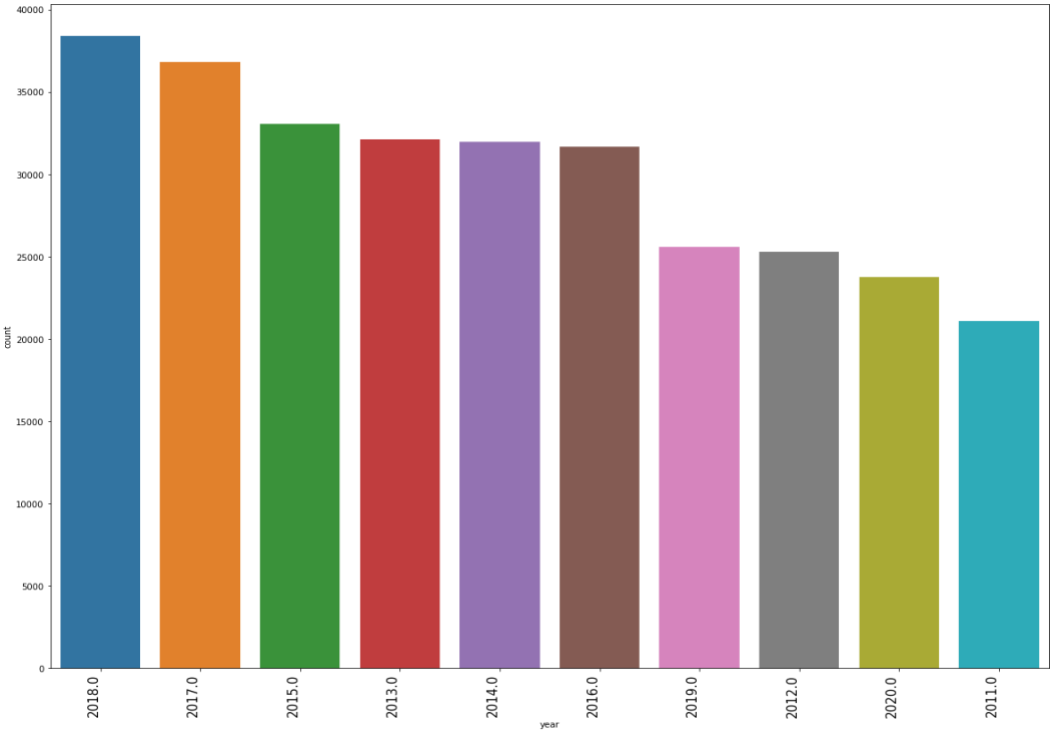


Figure 2.19 Higher car make years

- While doing so, another issue arose, which is that the condition labeled on some cars was not matching with their make year. Therefore, tinkering was done to the condition column values for appropriate categorization of the records as following while Fig. 2.20 displays the code that was used. The resulted outcome for condition-type car counts is displayed by the bar chart depicted in Fig. 2.21.
    - Made year > 2019 → "new"
    - Made year >=2017 & <=2018 → "like new"
    - Made year >= 2015 & <2017 → "fair"

```
dataset2['condition'] = np.where((dataset2['year'] >= 2019),
                                 "new",
                                 dataset2['condition'])
```

```
dataset2['condition'] = np.where((dataset2['year'] >= 2015) & (dataset2['year'] < 2017),
                                 "fair",
                                 dataset2['condition'])
```

```
dataset2['condition'] = np.where((dataset2['year'] >= 2017) & (dataset2['year'] <= 2018),
                                 "like new",
                                 dataset2['condition'])
```

Figure 2.20 Condition attribute tinkering code

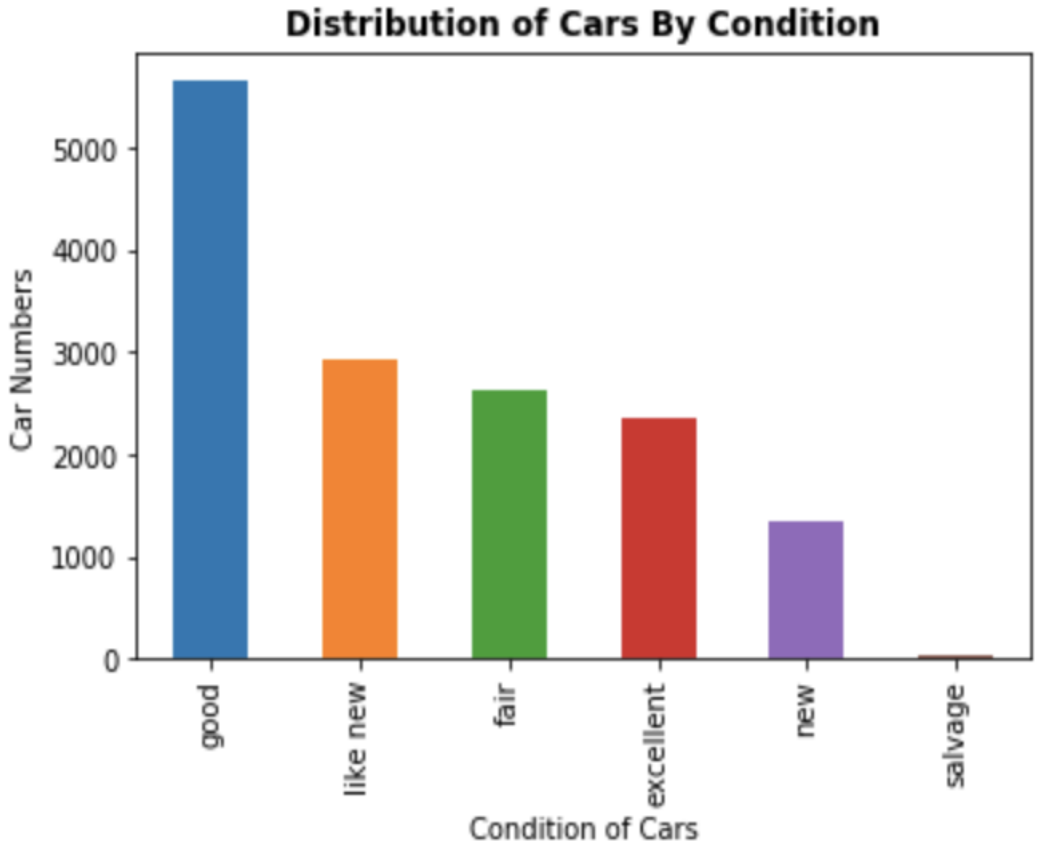


Figure 2.21 Distribution of cars by condition

- Finally, the last but not least process appeared at this point. That is the categorical variable conversion into numerical values by using the One-Hot-Encoding method, as shown by Fig. 2.22. It converts the textual data into either 1 or 0 values and is considered much more reliable in data representation without any issues, compared to the other method, which is Label Encoding that converts the categorical variable values into numbers in ascending order. Label encoding may infer a higher weight to a higher value and a lower weight to a lower value. That is why that method was disqualified from this research. Categorical encoding is obeyed since categorical variables cannot be used when regression-based ML algorithms are being involved, such as this own research. The numerically standardized or scaled dataset's first five-row view is depicted by Fig. 2.23, which now has 651 columns. At last, the cleaned dataset was exported into a new .csv format by the command shown in Fig. 2.24 and ended the data preprocessing process.

```
dataset2 = pd.get_dummies(dataset2)
```

Figure 2.22 Code for One-Hot-Encoding

price	year	odometer	manufacturer_chevrolet	manufacturer_ford	manufacturer_toyota	model_f-150	model_f-150 **1 owner clean	model_f-150 20" fuel wheels fuel	model_f-150 2dr single cab 4x4 v6 5 speed manual long bed	model_f-150 4wd	model_f-150 4wd supercab 145" xlt	model 150 4 superc 163"
20000	2011.0	101000.0	0	1	0	0	0	0	0	0	0	
15500	2004.0	60000.0	0	0	1	1	0	0	0	0	0	
6800	2008.0	202000.0	0	1	0	0	0	0	0	0	0	
3750	1978.0	123456.0	0	1	0	0	0	0	0	0	0	
3900	1997.0	230000.0	0	1	0	0	0	0	0	0	0	
22950	2014.0	166380.0	0	1	0	0	0	0	0	0	0	
10500	1988.0	10200.0	0	1	0	0	0	0	0	0	0	
12800	2000.0	97800.0	0	1	0	0	0	0	0	0	0	
8950	2011.0	164000.0	0	1	0	1	0	0	0	0	0	
31500	2019.0	106054.0	0	1	0	0	0	0	0	0	0	

Figure 2.23 One-Hot-Encoded categorical columns

```
dataset2.to_csv('cleanedData_1.csv')
```

Figure 2.24 Cleaned dataset export

2.3.3 Comparative study on 15 regressor models

A comparative study on 15 different regression algorithms was done, and those were implemented using the Scikit-learn ML library [18]. Each model was trained using the same training data and evaluated using the same testing data, and the train: test split ratio was 75:25 with 10 fold cross-validation to make the models less biased and account for every observation until satisfactory results emerge. Fig. 2.25 dispose of the python code used for the train and test dataset splitting with relevance to independent variables (x) and the dependent variable, price (y).

```
#separate the other attributes from the predicting attribute
X = dataset.drop('price',axis=1)
#separte the predicting attribute into Y for model training
y = dataset['price']
```

```
# importing train_test_split from sklearn
from sklearn.model_selection import train_test_split
# splitting the data
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Figure 2.25 Dataset train:test split

Moreover, overfitting, which is the level that the model fits exactly to a training dataset and result in poor predictions for unseen future observations while defeating the model's sole purpose, was eradicated to a commendable extent, enhancing generalization when K-fold cross-validation was utilized. In supervised ML, the regression-based method rather than the classification route has proven to be highly reliable and convenient in evaluating a continuous target variable [7]. After a comprehensive initial investigation, the most eligible 15 regressor models were identified to continue with the rest of the methodology for creating the optimum price prediction model, and below Table. 2.3 contextualize those models with a short introductory description.

Table 2.3 Comparatively studied regression algorithm models

Algorithm	Description
Extra Trees Regressor	Random forest-related decision tree bootstrap aggregation (bagging) ensemble included algorithm which performs better with few hyperparameters and sensible heuristics for configuration. The overall prediction is given by averaging the predictions of all decision trees. The whole training dataset is fitted in each decision tree. It is a better version than the Random Forest regressor [19].
Bagging Regressor	Decision tree-based model that does bagging ensemble similar to Random Forest. Combine the predictions of the decision trees by averaging. Each decision tree is fitted on a slightly different training dataset. Due to low correlated predictions and thereby errors, this model is quite effective [20].
Random Forest (RF) Regressor	Random Forest itself can be categorized as an ensemble model. It can be used for classification or regression type problems and contains a cluster of decision trees, either in hundreds or thousands in number, and goes very deep in training each tree separately. Researcher Ho has developed this particular algorithm to address the overfitting problem of decision tree algorithms overall [21].
Gradient Boosting Regressor	Simply an efficient algorithm that converts poor predictions into stronger predictions by accumulating misidentifications and errors that occurred in previous weak learners and adding stronger learners on the go. The three elements are a loss function, a weak learner for predictions, and an additive model to minimize loss function [22].

Decision Tree Regressor	The model is tree-structured. Dataset is broken down sample by sample while assigning decision trees for each. The final result would contain decision nodes and leaf nodes. Decision nodes contain one or more branches representing a value of an attribute, while leaf nodes contain the decision on a numerically aimed value. Both regression and classification type problems can be handled [23].
Multi-layer perceptron (MLP) Regressor	A feed-forward artificial neural network or a network that amalgamate multiple perceptron layers. The algorithm has three layers such as input, hidden, and output layer. Hidden and output layers use a linear activation function to conduct supervised learning through the backpropagation technique regarding training. Linearly inseparable data can be distinguished using this activation function and multiple layers, unlike linear MLP [24].
AdaBoost Regressor	AdaBoost is another boosting ensemble technique responsible for creating a strong regressor model from weaker models. Boosting can easily be understood by making this algorithm the starting point. An initial model is first created from the training dataset, and the second is built by rectifying the previous model errors. So on the sequence, proceed until a limit is achieved on added models or train data is predicted accurately. Decision tree performance can be boosted by using AdaBoost as well [25].
K-Nearest Neighbors Regressor (KNN)	It is basically a parameter-less classification model, and for input, k-closest samples are used for the training dataset, while the output will depend on whether the concerned issue is of a classification or regression type. Object's property value is determined for regression, and it is simply the averaged amount of the k nearest neighbors. Weights are assigned for neighbor contribution by distance, although a peculiarity is a sensitivity for local data structures [26].
Ridge Regressor	It is a special Tikhonov regularization developed as a solution (as an estimator) for least squared imprecision since most linear regression models retain multicollinear. This algorithm aims to estimate the coefficient of multiple linear regression where independent variables are strongly related. A more precise estimate of parameters is provided while variance and mean square estimator values are lower [27].
Multiple Linear Regressor	A statistical model is also known as multivariable regression, which extends ordinary least-squares (OLS) regression and is used when multiple explanatory variables and one target variable are present, and their inter-related relationship is in need of analysis. Most commonly used regression algorithm, especially when more than one predictors are present [28].

Lasso Regressor	A ML based regression analysis known as least absolute shrinkage and selection operator, that assist in variable identification and standardization in order to ameliorate the prediction power and interpretability of a model. It concurs that coefficient estimates do not need to be unique under the assumption of collinearity. Related to linear and ridge regression plus basis pursuit denoising [29].
Elastic Net Regressor	It is actually a regularized statistical and regression model that combines the L1 and L2 penance of lasso and ridge regression. It is known to be reduced and applied to linear support vector machines as well. Lasso is outperformed by Elastic Net with a similar representation and is not much useful when predictors are much higher than the observations [30].
Stochastic Gradient Descent (SGD) Regressor	As ensemble bagging and random forest algorithms allow decision trees to be greedily produced with training dataset subsamples, this objective function optimizing regressor helps to reduce the correlation between consecutive boosting models. That means a randomly selected sample fits as the base learned instead of the complete set. Faster iteration, as opposed to lower convergence, is achieved [31].
Support Vector Regressor (SVR)	This algorithm can be incorporated for solving both classification and regression-type algorithms. It can be easily trained to tag input data with labels, to divide them in the widest area possible, between categories. In the event of label unavailability, SVM cannot be applied for it. It requires applying another version of SVM, known as Support Vector Clustering, which is an unsupervised method, to categorize unlabeled data [32].
Linear SVR	The Linear SVR is a regression problem-solving algorithm based on the support vector machines technology. The linear kernel approach is used in the Linear SVR algorithm, which works well together with huge datasets. In this model, the L1 or L2 technique can be expressed as a loss function [33].

All necessary dependencies were imported first and foremost to start the training, and testing procedures of these models with the same cleaned used car dataset exported.

Feature scaling (normalization) from StandardScaler() was used to deal with KNN, SVR, linear SVR, and SGD regressors, as shown by Fig. 2.26.

```python
from sklearn.preprocessing import StandardScaler

#Mean variance normalization
standardscaler = StandardScaler()
standardscaler.fit(X_train)
X_train_std = standardscaler.transform(X_train)
X_test_std = standardscaler.transform(X_test)
```

Figure 2.26 Feature Scaling

When tuning the hyperparameters of the models and obtaining the optimum ones, the three major methods that were followed are **K-fold cross-validation**, **RandomizedSearchCV**, and **GridSearchCV**. By doing that, the model accuracy, speed, and prediction power increased, all the while reducing error rates. The relevant code showing the GridSearchCV applied to the MLP regressor is depicted by Fig. 2.27 while Fig. 2.28 displays RandomizedSearchCV inferred on RF regressor.

```python
mlp = MLPRegressor()
param_grid = {'hidden_layer_sizes': [i for i in range(2,20)],
              'activation': ['relu'],
              'solver': ['adam'],
              'learning_rate': ['constant'],
              'learning_rate_init': [0.01],
              'power_t': [0.5],
              'alpha': [0.0001],
              'max_iter': [1000],
              'early_stopping': [True],
              'warm_start': [False]}
mlp_GS = GridSearchCV(mlp, param_grid=param_grid,
                      cv=10, verbose=True, pre_dispatch='2*n_jobs')
```

Figure 2.27 GridSearchCV python code

```python
]: #no of trees
   n_estimators = [300, 350, 360, 370, 380]

   #max number of features to consider
   max_features = ['sqrt', 'log2', 'auto']

   #max number of levels in a tree
   max_depth = [10, 20, 30, 40, 50]
   max_depth.append(None)

   #minimal number of samples required to split a node
   min_samples_split = [2, 5, 10, 15, 20]

   #minimal number of samples required at each leaf level
   min_samples_leaf = [1, 2, 5, 10, 15]

   grid_param = {
       'n_estimators': n_estimators,
       'max_features': max_features,
       'max_depth': max_depth,
       'min_samples_split': min_samples_split,
       'min_samples_leaf': min_samples_leaf
   }

]: from sklearn.model_selection import RandomizedSearchCV
   RFR = RandomForestRegressor(random_state=1)
   RFR_random = RandomizedSearchCV(estimator=RFR, param_distributions = grid_param, n_iter=500, cv = 5, verbose = 2, ra
```

Figure 2.28 RandomizedSearchCV python code

In the course of tweaking the model parameters, one must be aware of each algorithm's types' capacity of performing, when a certain number of features are taken into consideration and when a complex dataset is imminent.

Custom arrays and functions were implemented to reuse the code when examining the models each by each and derive the R-squared value (R^2) (2), mean absolute error (MAE) (3), relative squared error (RSE) (4), and root mean square error (RMSE) (5) was used as accuracy criterion to determine the top 4 exemplary models out of all those considered.

$$R-squared = 1-\frac{\sum_i (y_i-\hat{y}_i)^2}{\sum_i (y_i-\bar{y})^2} \quad (2)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i| \quad (3)$$

$$RSE = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2} \quad (4)$$

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i-y_i)^2}{n}} \quad (5)$$

Fig. 2.29 depicts the custom arrays created for train and test dataset related accuracy metric storage and the two functions used to calculate the RMSE and RSE.

```
In [1]: acc_train_r2 = []
        acc_test_r2 = []
        acc_train_d = []
        acc_test_d = []
        acc_train_rmse = []
        acc_test_rmse = []
```

```
In [2]: from sklearn.metrics import r2_score
        def acc_d(y_meas, y_pred):
            # Relative error between predicted y_pred and measured y_meas values
            return mean_absolute_error(y_meas, y_pred)*len(y_meas)/sum(abs(y_meas))

        def acc_rmse(y_meas, y_pred):
            # RMSE between predicted y_pred and measured y_meas values
            return (mean_squared_error(y_meas, y_pred))**0.5
```

Figure 2.29 Specifically created arrays and functions

Fig. 2.30 illustrates the custom function ‘acc_model’ created to visualize each model’s accuracy estimators in a more comprehensible and coherent way while sectionizing the necessary resulted values for more readability.

```
In [3]: def acc_model(num,model,train,test):
            # Calculation of accuracy of model акщь Sklearn by different metrics

            global acc_train_r2, acc_test_r2, acc_train_d, acc_test_d, acc_train_rmse, acc_test_rmse

            ytrain = model.predict(train)
            ytest = model.predict(test)

            print('target = ', y_train[:5].values)
            print('ytrain = ', ytrain[:5])

            acc_train_r2_num = round(r2_score(y_train, ytrain) * 100, 2)
            print('acc(r2_score) for train =', acc_train_r2_num)
            acc_train_r2.insert(num, acc_train_r2_num)

            acc_train_d_num = round(acc_d(y_train, ytrain) * 100, 2)
            print('acc(relative error) for train =', acc_train_d_num)
            acc_train_d.insert(num, acc_train_d_num)

            acc_train_rmse_num = round(acc_rmse(y_train, ytrain) * 100, 2)
            print('acc(rmse) for train =', acc_train_rmse_num)
            acc_train_rmse.insert(num, acc_train_rmse_num)

            print('target_test =', y_test[:5].values)
            print('ytest =', ytest[:5])

            acc_test_r2_num = round(r2_score(y_test, ytest) * 100, 2)
            print('acc(r2_score) for test =', acc_test_r2_num)
            acc_test_r2.insert(num, acc_test_r2_num)

            acc_test_d_num = round(acc_d(y_test, ytest) * 100, 2)
            print('acc(relative error) for test =', acc_test_d_num)
            acc_test_d.insert(num, acc_test_d_num)

            acc_test_rmse_num = round(acc_rmse(y_test, ytest) * 100, 2)
            print('acc(rmse) for test =', acc_test_rmse_num)
            acc_test_rmse.insert(num, acc_test_rmse_num)
```

Figure 2.30 Model accuracy output visualization function

By utilizing the above-stated functions, a common procedure was followed regarding the models along with their own tuned hyperparameter values. Following pseudo code explains the overall methodology followed when training and testing the precision and error rates of the 15 different regression models individually.

```
PROCEDURE MODEL_TRAIN_TEST(regressor)
    BEGIN
        Import necessary dependencies
        Import Scikit Learn regressor models
        Import cleaned dataset
        Separate independent attributes and dependent attribute
        Dataset split into train_test at 75:25 (x_train, x_test, y_train, y_test)
        FOR each regressor model
        DO
            Model initialization with optimal hyperparameter values
            x_train and y_train fit into created model
            Call function acc_model(model_number, model, x_train, x_test)
        UNTIL all models executed
        END FOR
        Visualize final table having accuracy and error metrics of all models
    END
```

The results acquired and visualized in Jupyter Notebook regarding the models will be revealed in the 3rd chapter, 'Results & Discussion.' The ulterior motive of this phase was to ascertain the top 4 exemplary models for a further innovative approach, which is to apply a heterogenous ensemble learning technique to produce a combined price prediction. Since the individual analysis was done foremostly, a better perception of how the ensemble model's accuracy will differ in comparison to the singular models themselves was proclaimed.

2.3.4 Top 4 regressor infused ensemble model development

It could have been sufficient for some research done in the past to just identify the most optimum model and continue on with it further towards the final prediction result and application visualization. Although that is the most common case, this research intended to dive even further and examine how those selected successive models could improve their performance even more if they were grouped together. Ensemble learning can be known as a technique to combine and amalgamate different models to attain bolstered and enhanced prediction results, rather than when using them individually for the task. Although in the regressor evaluation phase, the hyperparameters were tuned properly using various techniques, and K-fold cross-validation was done to decrease the overfitting capacity by navigating on the ensemble

path, both model optimization (provide correct predictions) and generalization (how it can perform for unseen data) can be further ameliorated. The following mentioned techniques can be used to develop hybrid models with high accuracy and value [34].

a) Majority voting
b) Weighted voting
c) Simple averaging
d) Weighted averaging
e) Bagging
f) Boosting
g) Stacked Generalization

The above-stated a) to d) methods are the most easily understood ensemble approaches with their verbose itself, and the voting method is used to solve classification problems while averaging is used in the regression. On the other hand, e) to g) are ensemble techniques that promote meta-learning algorithms to produce a hybrid model that can reduce variance (through bagging), decrease biasing effect (through boosting), and bolster the generalization in prediction (through stacking). Sequential and parallel ensemble methods are the two mainly divided groups. In sequential ensemble methods, they are created sequentially, having the dependence between each other to exploit a boosted performance by weighing previous weighted misinterpretations (Boosting), whereas parallel method base learners are developed parallelly while giving priority to learner independence to reduce error rates drastically through averaging (Bagging). Bagging uses a set of independent trees to predict an outcome. Boosting methods use sequential iterative learning and conversion of weak models into powerful ones through found faults in precedent models contain homogeneous, or in other words, single base learning algorithms. Random forest and gradient boosting are the two most infamous examples for bagging and boosting, respectively. Below Fig. 2.31 and Fig. 2.32 shows the basic structure of how those two bagging and boosting algorithms are structured at a tree level when compared with a single decision tree [35].

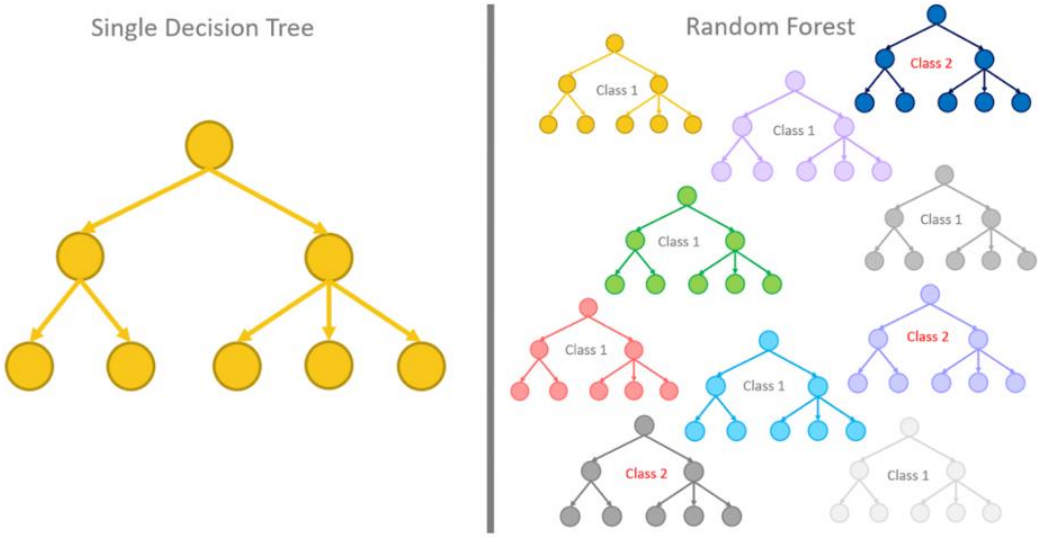


Figure 2.31 Random Forest bagging structure [35]

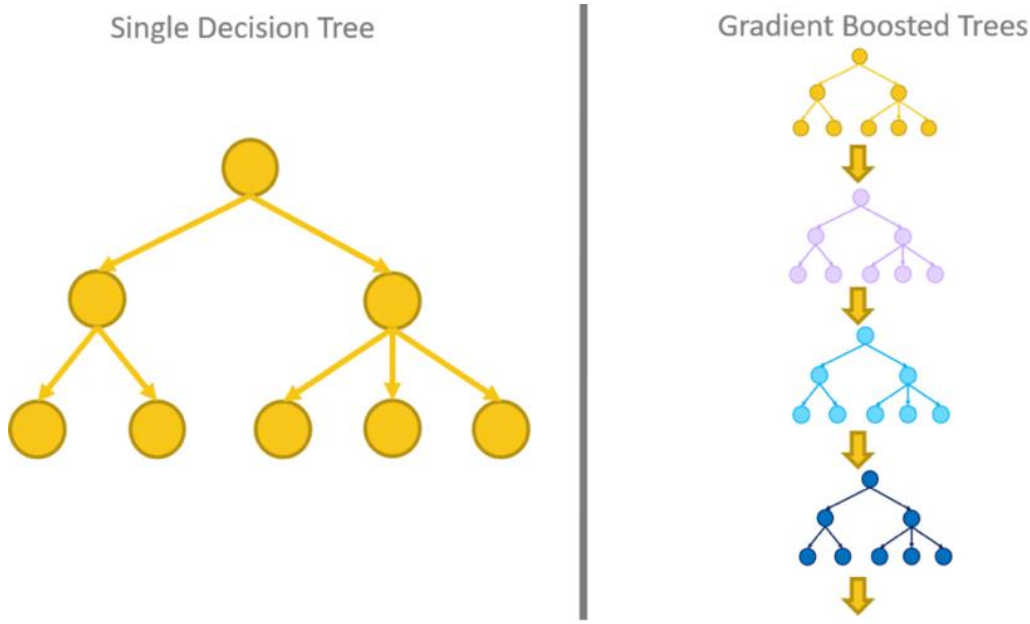


Figure 2.32 Gradient Boosting ensemble structure [35]

This research conducted its ensemble experiment using the stacked regressor/stacking generalization method with the triumphantly functioning four heterogenous regressors that were inaugurated after the comparative study. The reason for this choice was being able to experience the functionality of both bagging and boosting ensembles through the pronounced regressors since the majority consisted of such ensemble-based algorithm structures.

To provide an accurate prediction by the assembled model than any of its individual models, the base learners are required to be as distinctive in nature and precise as possible, and the sophisticated analysis on the 15 regressors provided expediency for that. The stacking method includes base learners and a meta-leaner(meta-regressor), and the feature outputs of the trained (with the thorough training dataset) base learners are obtained by the meta-model for its own training and final forecast render. Following Fig. 2.33 depicts the algorithm used for the stacked regressor ensemble learning process.

Algorithm Stacking

1: Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
2: Ouput: ensemble regressor H
3: *Step 1: learn base-level* regressor
4: **for** $t = 1$ to T **do**
5: learn h_t based on D
6: **end for**
7: *Step 2: construct new data set of predictions*
8: **for** $i = 1$ to m **do**
9: $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_T(x_i)\}$
10: **end for**
11: *Step 3: learn a meta-* regressor
12: learn H based on D_h
13: return H

Figure 2.33 Stacking algorithm [36]

The hyperparameters utilized for the optimum accuracy rate regarding the individual models are shown in the 3rd chapter, and the same were used when they were to be combined together to form the stacking regressor. The adjusted R^2 value was analyzed for the top for individual models to ascertain their proven strong prediction power.

```
In [10]: #Validation function
         n_folds = 10

         def rmse_cv(model):
             kf = KFold(n_folds, shuffle=True, random_state=42).get_n_splits(x_train.values)
             rmse= np.sqrt(-cross_val_score(model, x_train.values, y_train.values, scoring="neg_mean_squared_error", cv = kf)
             return(rmse)
```

```
In [22]: # get a stacking ensemble of models
         def get_stacking():
             # define the base models
             level0 = list()
             level0.append(('bagging', BaggingRegressor()))
             level0.append(('extraTree',ExtraTreesRegressor(n_estimators=100)))
             level0.append(('randomForest', RandomForestRegressor(max_features = 'sqrt', n_estimators = 360, random_state = 4,
             # define meta learner model
             level1 = ensemble.GradientBoostingRegressor(n_estimators=15000, learning_rate=0.05,max_depth=4,max_features='sqrt
             #define the stacking ensemble
             model = StackingRegressor(estimators=level0, final_estimator=level1, cv=5)
             return model
```

```
In [23]: et a list of models to evaluate
          get_models():
          models = dict()
          models['bagging'] = BaggingRegressor()
          models['extraTree'] = ExtraTreesRegressor(n_estimators=100)
          models['randomForest'] = RandomForestRegressor(max_features = 'sqrt', n_estimators = 360, random_state = 4, min_sam
          models['gradientBoosting'] = ensemble.GradientBoostingRegressor(n_estimators=15000, learning_rate=0.05,max_depth=4,
          models['stacking'] = get_stacking()
          return models
```

```
In [24]: # evaluate a given model using cross-validation
         def evaluate_model(model, X, y):
             cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
             scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1, error_score='raise')
             return scores
```

Figure 2.34 Stacking regressor custom functions

The custom codes written in Jupyter Notebook for the requirements of building up the stacking regressor are shown in Fig. 2.34 above. The mean and standard deviation of the prediction scores of both individual models and the stacked regressor model is then analyzed and displayed through a boxplot for clear visualization. These results will be exhibited in the 3rd chapter for results and discussion as well.

2.3.5 Ensemble model serialization

After the analysis and further refinements, the ensemble model was to finally be serialized (exported from the python file in order to be used by an external API or any other server for storage purposes) using the python library known as 'Pickle'. Pickle was chosen rather than the joblib library because of its convenient ability to transform any python object (dict, list), which is complex, into a byte stream and vice versa. Since pickle was old enough coming far back from python version 1.4, making it even more mature than pip itself, it encouraged its incorporation within thin research component implementation to attain stability and dependability, having sound

familiarity and many exertions within the domain of ML algorithms. Below Fig. 2.35 diagram depicts the code used for the model serialization task.

```
In [67]: # Save/serialize stacking model
         import pickle

         pickle.dump(bagging,open('stackingModel.pkl','wb'))
         print("Stacking regressor ensemble model dumped successfully!s")
```

Figure 2.35 Pickle model dump

Since the column heading values need to be imported as well towards the API for the prediction of the price according to the newly submitted car advertisement details, they need to be serialized to a character stream by pickle as well, especially considering the fact that the column headings have been changed from its original due to OHE conversions. The same nature must be acquired by the API for function executions. Fig. 2.36 displays the relevant python code for that task.

```
# Saving the data columns from training
model_columns = list(X.columns)
pickle.dump(model_columns, open('model_columns.pkl', 'wb'))
print("Models columns dumped!")
```

Figure 2.36 Pickle model column dump

2.3.6 Flask API implementation & model deserialization

The next step of the price prediction system implementation was to develop and integrate the ML API built from the Python microframework Flask. The reason for selecting Flask for the experimentation of building an ML model facilitating API is due to its lightweight structure as minimalistic features without having to be dependent on external libraries or other frameworks. The ML model functionality is complex itself; hence another sophisticated framework was unnecessary to be incorporated, considering performance, resource consumption, and latency metrics. Flask may only and mostly depend on the Jinja engine (template) and additionally on the Werkzeug WSGI toolkit. Nonetheless, its' lightweight codebase and easy implementation/maintenance was the prominent, influential factor considered. Below illustrated Fig. 2.37 shows all the python related dependencies imported within the priceApi.py file implemented API. The .pkl files containing the stacked regressor model and the column headings were deserialized using pickle.load() function.

```python
# Dependencies
from sklearn.base import BaseEstimator, TransformerMixin, RegressorMixin, clone
from flask import Flask, request, jsonify, render_template
import pickle
import sys
import traceback
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from flask_cors import CORS
app = Flask(__name__)
cors = CORS(app)

#   Load related columns
model_columns = pickle.load(open('model_columns.pkl', 'rb'))

# Load models
stacking_regressor = pickle.load(open('stackingModel.pkl', 'rb'))


@app.route('/')
def home():
    return render_template('predict.html')
```

Figure 2.37 Flask API part 1

Additionally, a middleware with the name of Cross-origin Resource Sharing (CORS), which is an HTTP-header based protocol that enables and permits a browser running client-side application to access resources from different external origins (domain, ports, schemes), was installed using 'npm' command and imported into the Flask API. That was done in order for the front-end application to send API targeted requests without facing any security issues and allowing the permission of direct specific API endpoint access. Hence the API became a 'cors' app. Below exhibited Fig. 2.38 displays the rest of the code in the API, which indicates the function targeted by a POST type request towards the hosted server's "/predict-price" route to acquire the car advertisement attributes as a JSON string, convert it to a data frame and do the necessary encoding by the pd.get_dummies() method prior to being attached with the OHE column headings and final prediction using the stacked regressor model which was deserialized.

```python
@app.route('/predict-price', methods=['POST'])
def predictPrice():
    if bagging and etr and randomForest and gradientB:
        json_ = request.json
        df = pd.DataFrame(json_, index=[0])
        print(df)
        query = pd.get_dummies(df)
        query = query.reindex(columns=model_columns, fill_value=0)
        print(query)

        model = (stacking_regressor)

        stacked_regression_predictions = np.column_stack([
            model.predict(query) for model in model
        ])

        final_prediction = np.mean(stacked_regression_predictions, axis=1).round(2)
        price_range1 = (final_prediction - 3000).round(2)
        price_range2 = (final_prediction + 3000).round(2)

        # perform conversion
        final_prediction = str(final_prediction)[1:-1]
        price_range1 = str(price_range1)[1:-1]
        price_range2 = str(price_range2)[1:-1]

        return jsonify({'price': str(final_prediction)})

    else:
        print('Train the price prediction model first')
        return 'No price prediction model deployed here to use'


if __name__ == '__main__':
    app.run(host='localhost', port=2096, certfile='cert.pem', keyfile='key.pem', debug=True)
```

Figure 2.38 Flask API part 2

The code is structured in a flexible way so that the final prediction can be brought up as an exact price that was predicted or as a price range as well. Regardless, the final price outcome will be sent as a JSONIFY response into the client-side application for customer visualization purposes. The bottom method signifies the API booting process having the designated host port of 2096, the cert file, and the key file necessary for cloud server deployment.

2.3.7 API & model deployment in cloud server

Succeeding the triumphant API implementation with the integration of the developed ML ensemble model, it was necessary for it to be deployed in a cloud server environment for the client application's efficient accessing. For that, the DigitalOcean

Infrastructure as a Service (IaaS) cloud service was utilized. The cloud environment supports robust application development with simplicity being merged. Scaling is made easy by allowing for the applications to be deployed on multiple servers if necessary.

The droplet which was purchased from the DigitalOcean has an NGNIX proxy along with a Supervisorctl that helps to restart any service or API which has been failed or stopped. Supervisorctl has the ability to continuously monitor for occurrences and prevent them from happening or undertake mitigating actions. Load balancing is additionally included within the server functionality, which can be known as another unique factor of it.

The server cost monthly is $ 27.21, and the main specifications are listed below.

- CPU – 2
- RAM – 2 GB
- Transfer – 3 TB
- SSD Disk – 60 GB

Inside the droplet, all research members' components are deployed together to enhance communication and maintenance facilities. Having a single point of access is deemed better regarding a research project of the current stage, and since it is a team effort to build the overall application, all members are able to manage the cloud resources as necessarily where monitoring the changes done by the others.

Even though the price prediction API and the ensemble model with the dependent .pkl files are deployed together with other extensive, profuse member ML algorithms, the performance and prediction accuracy are not disturbed in any considerable way. Fig. 2.39 shown below convey the dashboard view of the purchased droplet (server) from DigitalOcean under the specific folder structure having the application name 'Tievs.' The login credential access belongs to the group leader of this research project. A total of 3 droplets of the same specification has been acquired for the deployment of the member ML components and the front-end application.

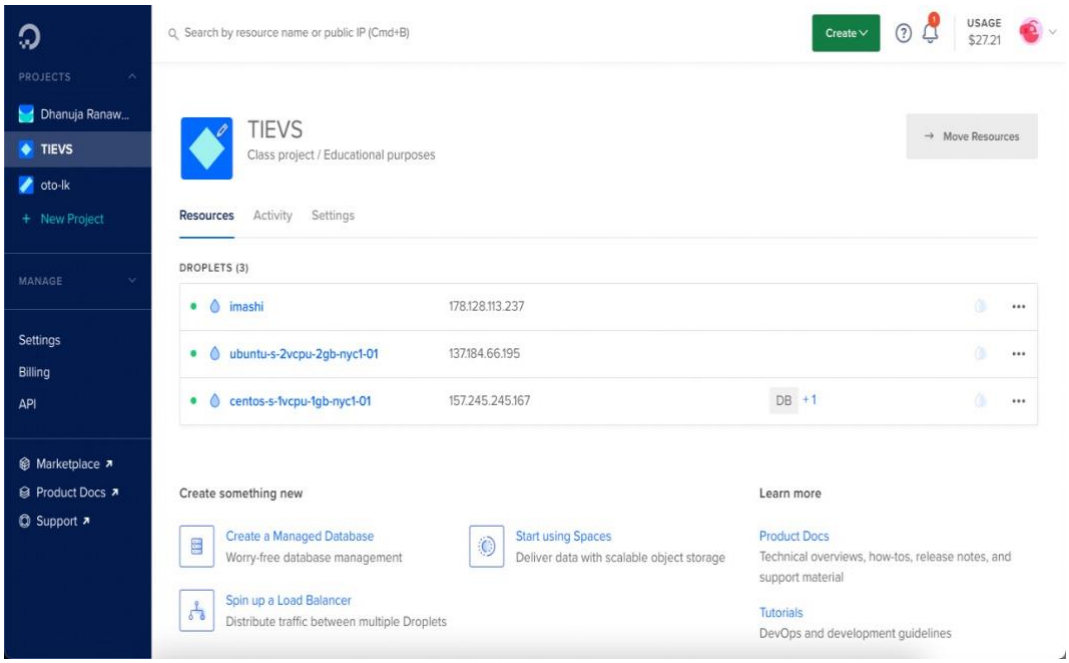


Figure 2.39 DigitalOcean droplet dashboard

Moreover, below shown Fig. 2.40 demonstrate the specific droplet's DigitalOcean UI view of which the price prediction API and ensemble model with dependencies reside.

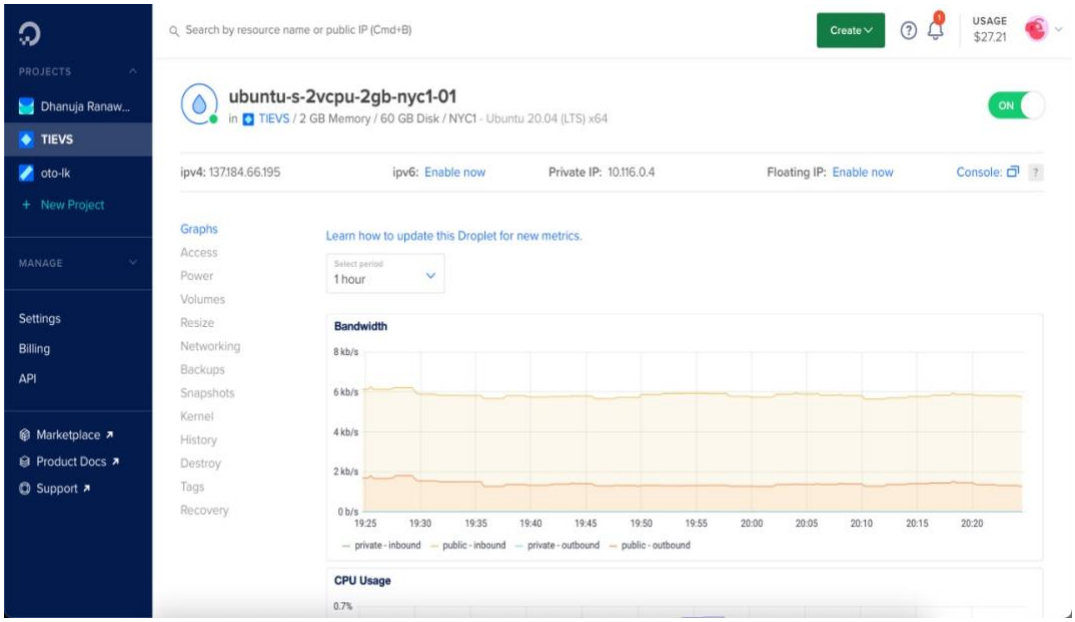


Figure 2.40 Specific droplet dashboard UI

The next Fig. 2.41 displays the command line view of the Secure Shell (SSH) tunnel of the server purchased. This view is being used to manage the server's internal

processes using Ubuntu commands and to monitor the memory, CPU process usages from time to time. Referring designated documentation can also be done by entrusting this console.

```
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-84-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Sep 30 14:55:22 UTC 2021

  System load:  0.0                Users logged in:       0
  Usage of /:   47.9% of 57.98GB   IPv4 address for eth0: 137.184.66.195
  Memory usage: 47%                IPv4 address for eth0: 10.10.0.7
  Swap usage:   0%                 IPv4 address for eth1: 10.116.0.4
  Processes:    134

83 updates can be applied immediately.
To see these additional updates run: apt list --upgradable


*** System restart required ***
Last login: Mon Sep 27 18:41:35 2021 from 112.134.175.138
root@ubuntu-s-2vcpu-2gb-nyc1-01:~#
```

Figure 2.41 Ubuntu server SSH tunnel

Above mentioned Supervisorctl monitoring view from the command line interface perspective is illustrated by Fig. 2.42 below. As displayed, the rest of the member APIs deployed are being analyzed for any fault processes as well since all are located in the same server location for facile scrutinizing.

```
PriceP — root@ubuntu-s-2vcpu-2gb-nyc1-01: ~ — ssh root@137.184.66.195 — 122×39
root@ubuntu-s-2vcpu-2gb-nyc1-01:~# sudo supervisorctl status
fraudtxt-api                     RUNNING   pid 112502, uptime 2 days, 20:06:45
km-api                           RUNNING   pid 112503, uptime 2 days, 20:06:45
price-api                        RUNNING   pid 112504, uptime 2 days, 20:06:45
recsystem-api                    RUNNING   pid 145659, uptime 5:57:22
root@ubuntu-s-2vcpu-2gb-nyc1-01:~#
```

Figure 2.42 Status of the Supervisorctl

2.3.8 Front-end application development, integration & deployment

The front-end application, dubbed Tievs, is built primarily on the Angular framework, with Typescript serving as the programming language for UI components. It's another open-source, free web design and development framework created by Google's Angular team and other independent volunteer groups, aimed mostly for single-page client-side applications, which is much more practical for Tievs as an e-commerce web app. During implementation, the bedrock and optional facilities are imported as typescript libraries. Because it can generate reusable, accurate, and maintainable code with universality and support for JavaScript XML (JSX), Typescript is suited for statistically highly paced front-end development contexts. Aside from safety, source code architecture is also a bonus aspect. When using Angular, specific back-end capabilities to produce various services and API invocating operations is not required because it is not just a set of libraries but a fully developed framework, which is why this research constituent chose Angular and misapply React, which is mostly a library that requires back-end support.

To make the UI features more attractive and eye-catching towards the consumers, Material UI designing dependent libraries for Angular were combined with basic HTML and CSS. Additionally, Angular incorporated services are in charge of database and API function calls from the front-end application Tievs itself, while the components manage the UI characteristics and their vibrant, precise replies without the need for repeated page reloading. It essentially acts as a back-end solution without the need for any further coding.

The Tievs front-end application is being deployed in the cloud environment known as Cloudflare pages. It is based on JavaScript and transposable API with HTML markup, which in some instances can be known as a JAM stack deployment and collaboration platform as well. Cloudflare provides dynamic and effort-free features for developers who are aiming for an unrivaled speed. The Cloudflare dashboard, which analyzes the Tievs client application visitors is being depicted by Fig. 2.43.

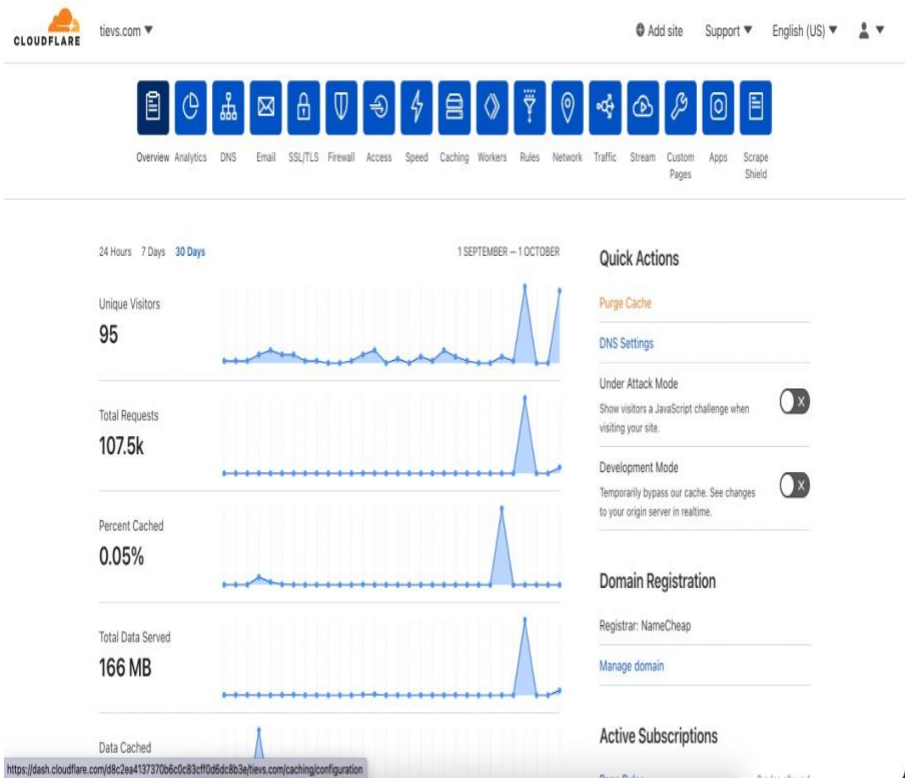


Figure 2.43 Cloudflare Tievs dashboard part 1

Metrics such as unique visitors, total requests, percent cached, total data saved, and cached can be graphically presented and viewed from the above dashboard whereas the below Fig. 2.44 shows the second part of the Cloudflare dashboard where production platform related status and statistical information is being displayed.

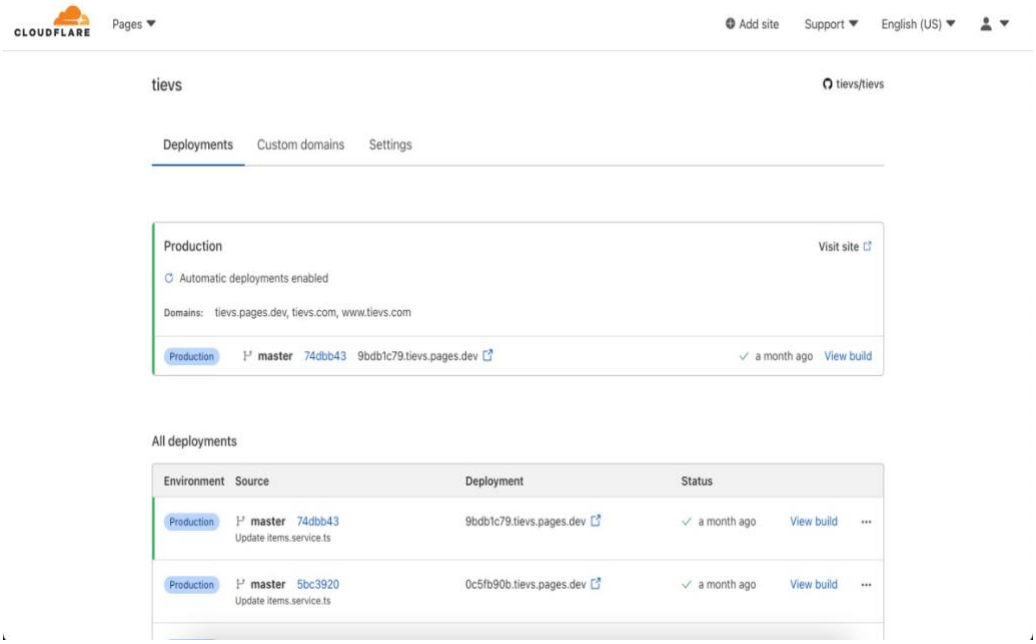


Figure 2.44 Cloudflare Tievs dashboard part 2

Successful deployment of the front-end application led to the progressive integration of the ML API and the ensemble model in order for the customer to explore the web application on preferred classifieds, and then if needed, click on the 'Post Ad' button to continue on with the filling of the new classified ad submission by accordingly entering the required characteristics for vehicle price prediction. With fruitful attainment of this objective, the road to price prediction and recommendation of it to the end-user will be achieved successfully.

The front-end UI result of the Tievs home page, single classified advertisement view, and the section where the consumer must fill out the necessary car attribute to facilitate the price suggestion in a user-friendly manner which allure customer observation, will be displayed in the 3rd chapter 'Results & Discussion'.

The below diagram depicted by Fig. 2.45 shows the basic structure of how the ensemble model, the price prediction API, the front-end Tievs' application's advertisement submission form UI, and the end-user/customer establish the essential networks and/or communications.

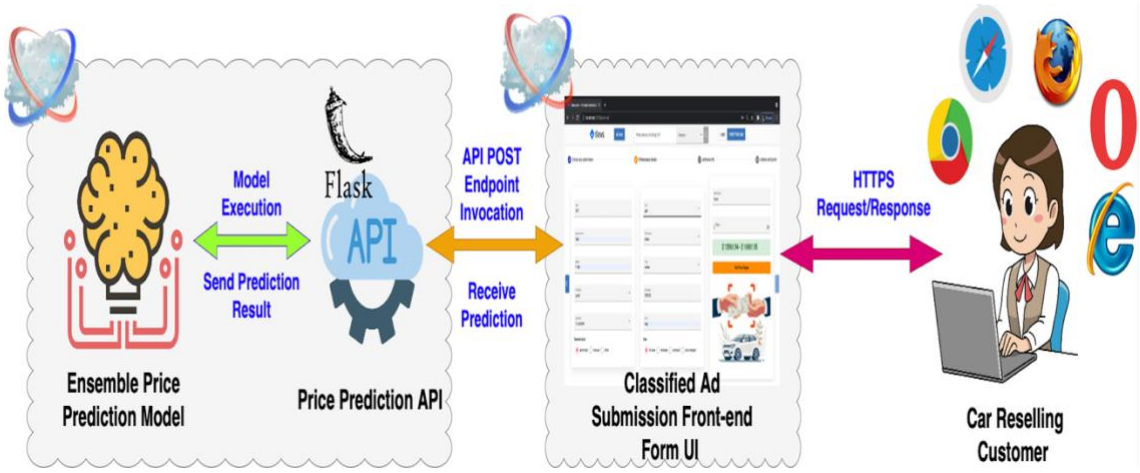


Figure 2.45 Communication from prediction model to customer & vice versa

2.4 Testing

2.4.1 Functional testing

Testing is of the highest significance when it comes to the Agile software development life cycle. Continuous and repeated tests were done during the implementation phase in all sections, from data engineering to front-end application deployment. Tests were executed in parallel to development work and not only near system completion. Unit testing, smoke testing, regression testing, component testing, integration testing, API testing, UI testing, black-box testing, and production testing were some of the functional testing types conducted during the implementation of the price prediction system and Tievs application as a whole. Some high-level test cases that were executed can be viewed from Appendix A. API testing of the hybrid model integrated Flask API was done almost nearing the end of the cycle to clearly and accurately evaluate the authenticity of predicting a price value. This black-box type testing procedure aims to validate the research components' conformance to its initial functional specifications and requirements.

The main function that is happing and would need to be tested is the model and API execution which provides a valid price prediction result. By testing the API POST endpoint directly, the above would be satisfied. That way, an assertion could be attained that the ensemble model is performing as required and the API endpoint is being accessed without facing any errors. For this purpose, Postman was chosen because of its ease of use in testing HTTP/2 (HTTPS) requests through a symbolical UI that satisfies all of the requirements for any type of request parameters, variously formatted request/response bodies, and request types such as GET, DELETE, POST, and PUT. Different forms of responses can be obtained, and they can all be validated. Requests can be kept and reviewed using collection categorization and traffic process estimators like status responses, time, and size. Furthermore, using Postman as an HTTP client allowed for the administration of authorizations, scripts, and headers with regard to the API to be tested. Below Fig. 2.46 evince the Postman UI of the test case for assessing the POST request sent towards the following deployed API endpoint.

The request parameters are sent as raw JSON formatted strings having the necessary key and value pairs relating to a car's features. A total of 13 attributes are being stated and sent. A response is then received indicating the predicted price from the ML ensemble model, that too as a JSON string. The response status is shown as 'OK' with the code of 200, and its' size is stated as 659B, which is considerably low and healthy for the request calls. The overall time from the request to the response by accessing the cloud environment hosted is 3.38 seconds without accounting for caching. With caching, the response time was reduced to 410 milliseconds.

API POST Endpoint – https://ml.tievs.com:2096/predict-price

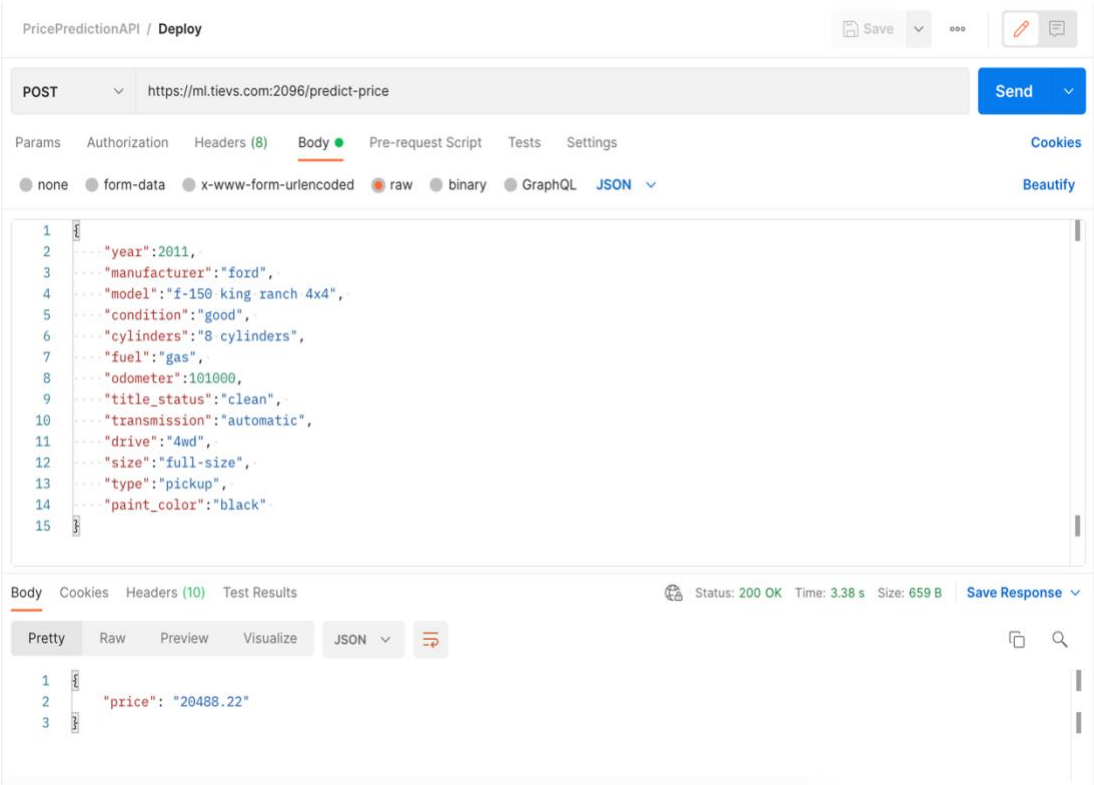


Figure 2.46 Postman price prediction API test

The testing was done twice, which is with an older version of the dataset, and then again after obtaining an updated version of the used car dataset from the same online location. The latter testing phase results were satisfactory, and the updated dataset was responsible for the training of the ML models and the ensemble model afresh. All implementation details and results containing in this thesis is of the newly updated dataset trained ML models and stacking regressor model.

2.4.1 Non-functional testing

Not only the functional aspect of this research component but also the non-functional requirements were also contemplated by this experiment. Consumers accessing the services from the price prediction model should have accurate and reliable information received within a small amount of time. The performance statistics of the API endpoint were tested considering that factor. Performance testing is a huge domain when discussing non-functional testing. It helps to assess the system under test for its speed, stability, availability, response time, and reliability for varying loads of user sessions. Load testing was performed in this research to test the ML API endpoint performance by stimulating 5 users at a single time to monitor and verify the server behavior under a traffic load.

Another advantage of conducting the load test, which is a part of performance testing, is to examine how the ML model integrated API located server itself performs under high and low fluctuating loads. This way, it was easier to determine the adequacy of the server droplet that was purchased as well. However, it was not necessary to change the already acquired droplet since the server could handle the load effortlessly. For the purpose of conducting the load test for this research component, Loadmill was utilized. It is an automated testing tool that most developers love and enjoy working with, especially to test their cooperate client servers. This behavior-driven solution has a rich user interface to displays various performance graphs and tables. It is responsible for storing various collections of test cases as well so that the developers could refer to them later on and continue on with the testing procedures. Main CI/CD challenge is to test the newly updated codes without dragging behind schedule, and Loadmill provides the best platform for that with its test creation technology and swift feedback loop, which detects bugs in source code. In addition, it has the ability to transform realistic user behavior into testing procedures automatically, along with the consideration of the API contexts and correlations. Hence, why this research auditioned the same tool for its load testing needs. Following Fig. 2.47 and Fig. 2.48 depicts the Loadmill load test results presented in an attractive graphical view. The total elapsed time for the test was 2 minutes and 3 seconds. The active simulated user

sessions were 5 between each second in the load test throughout. It gained an error rate of just 1.06%, which is an extremely outstanding performance. Normally, it is stated that these types of ML model integrated APIs perform quite poorly when it comes to the error rate statistic, although this price prediction API under test begged to differ in view. Hence it is definitely a remarkable result. The average response time calculated was 501 milliseconds which can be considered as another astounding outcome received with regard to an efficient response time, which many researchers try to attain by their built ML model incorporated APIs. Moreover, the performance over time and the throughput in requests per second (rps) are graphed for convenient visualization.

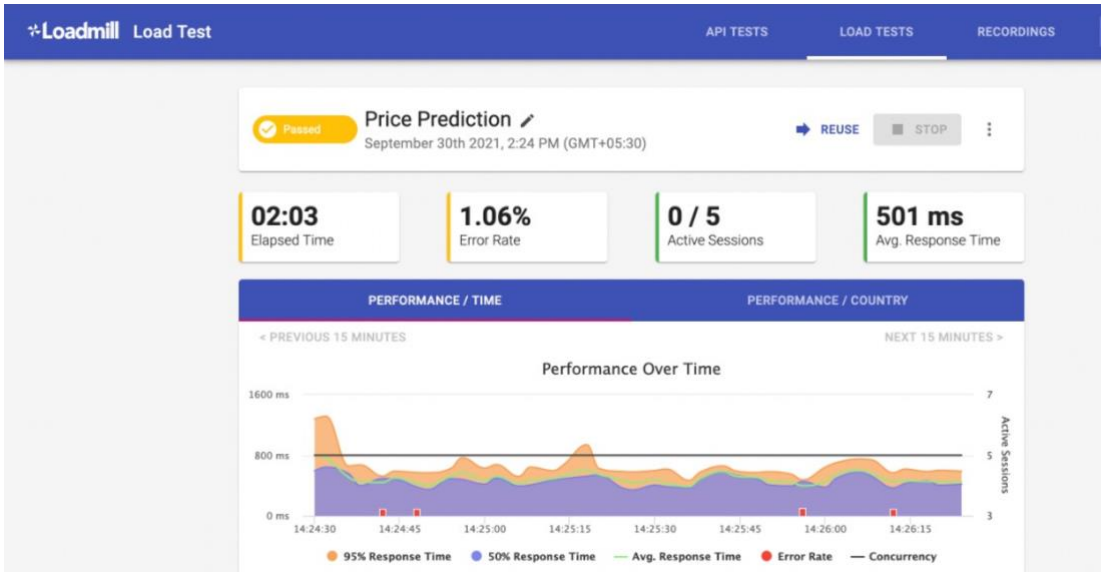


Figure 2.47 Price prediction Loadmill test part 1

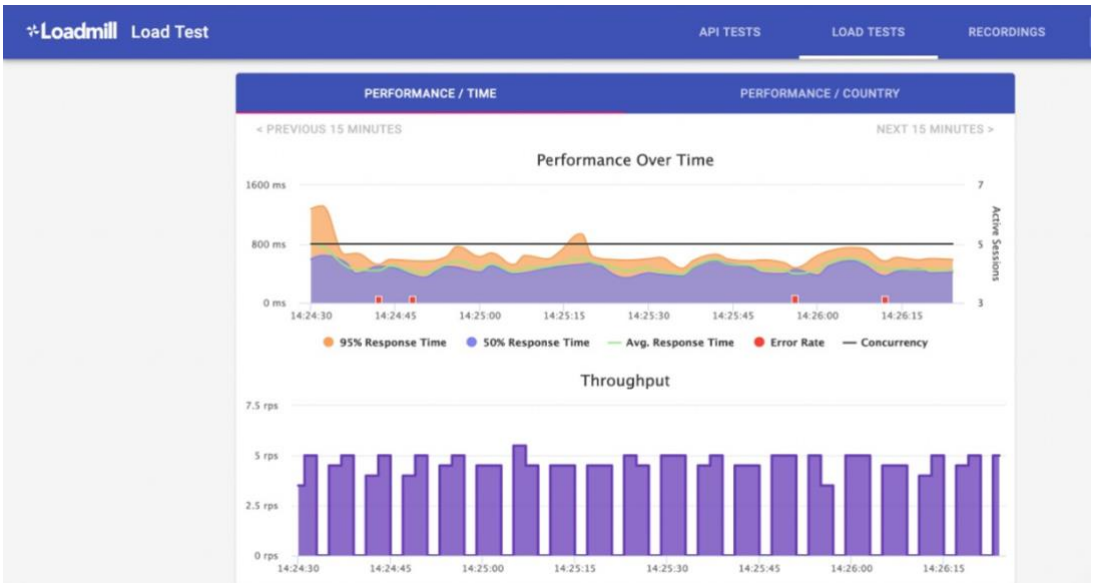


Figure 2.48 Price prediction Loadmill test part 2

3 RESULTS & DISCUSSION

3.1 Results

One of the most salient results found is from the correlative analysis proceeded with the selective fifteen regressors. The most optimum hyperparameters were used in each model prior to fitting the model with the training dataset. Below are the initialization code used to create the Scikit learn library imported 15 regressor models before the training and testing started.

```
1. etr = ExtraTreesRegressor(n_estimators=100)
2. abr = AdaBoostRegressor(learning_rate=0.1)
3. bagging = BaggingRegressor()
4. decision_tree = DecisionTreeRegressor()
5. gradientmodel = ensemble.GradientBoostingRegressor(n_estimators=5000,
   learning_rate=0.05,max_depth=4,max_features='sqrt',min_samples_leaf=15,
   min_samples_split=10,loss='huber')
6. knn_reg = KNeighborsRegressor(weights='distance', n_neighbors=10)
7. lasso_rg = Lasso(alpha=0.1, max_iter=10000)
8. mlp_GS = GridSearchCV(mlp, param_grid=param_grid, cv=10,
   verbose=True, pre_dispatch='2*n_jobs')
9. sgd = SGDRegressor(max_iter=1000, tol =1e-3)
10. lr = LinearRegression(n_jobs = 20)
11. rf = RandomForestRegressor(max_features = 'sqrt', n_estimators = 360,
    random_state = 4, min_samples_split = 2, min_samples_leaf = 1, max_depth
    = 30, bootstrap = True)
12. rr = Ridge(alpha=0.01, normalize=True)
13. svr_rbf = SVR(kernel='rbf', C=5000, gamma=0.01,
    epsilon=.01).fit(x_train_stdS, y_train)
14. linear_svr = LinearSVR(max_iter=50000, epsilon=.01)
15. E_Net = ElasticNet(alpha=0.005, l1_ratio=.9, random_state=3,
    max_iter=6000)
```

It should be noted that some of the regressors were instantiated using their default hyperparameters as well since the RandomizedSearchCV and GridSearchCV methods declared those default values as the most effective. Furthermore, the param_grid values used for the MLP regressor created above are as below.

```
param_grid = {'hidden_layer_sizes': [i for i in range(2,20)],
              'activation': ['relu'],
              'solver': ['adam'],
              'learning_rate': ['constant'],
              'learning_rate_init': [0.01],
              'power_t': [0.5],
              'alpha': [0.0001],
              'max_iter': [1000],
              'early_stopping': [True],
              'warm_start': [False]}
```

The prediction accuracy and error rates with respect to performance criterion of the above-fitted models such as R^2, RSE, and RMSE were evaluated, and a decision was made on which models performed in the most valiant and robust manner with healthy accuracies and comparatively low faulty predictions.

Since all these models underwent K-fold cross-validation during training and testing, the tendency to overfitting, or else overly fit and match exactly against noises and remnant outliers of the training data, was reduced, and the models were made to predict new, unseen, and future data values with high strength for generalization.

By executing the below python code section demonstrated from Fig. 3.1, a table was generated containing all 15 regressors, and their train/test dataset incurred results for a better perceptive comparison to identify the best four regressors according to the above mention 3 estimators. The adjacent Fig. 3.2 depicts the produced table from the said python code, visualizing all regressor results in the specific forecasting fruition order.

```
In [24]: models = pd.DataFrame({
             'Model': ['Extra Trees Regressor', 'AdaBoost Regressor', 'BaggingRegressor',
                       'DecisionTreesRegressor', 'Gradient Boosting Regressor',
                       'K Nearest Neighbours Regressor', 'Lasso Regression',  'MLP Regressor', 'SGDRegressor',
                       'MultipleLinearRegression', 'RandomForestRegressor', 'RidgeRegressor', 'Support Vector Machines Regres
                       'LinearSVR','ElasticNet Regression'],

             'r2_train': acc_train_r2,
             'r2_test': acc_test_r2,
             'd_train': acc_train_d,
             'd_test': acc_test_d,
             'rmse_train': acc_train_rmse,
             'rmse_test': acc_test_rmse })

In [25]: pd.options.display.float_format = '{:,.2f}'.format

In [26]: print('Prediction accuracy for models by R2 criterion - r2_test')
         models.sort_values(by=['r2_test', 'r2_train'], ascending=False)
```

Figure 3.1 Python code for comparison table generation

Out[26]:

	Model	r2_train	r2_test	d_train	d_test	rmse_train	rmse_test
0	Extra Trees Regressor	99.84	89.61	0.23	8.81	56,364.77	447,910.22
2	BaggingRegressor	98.00	89.31	4.07	10.07	199,277.80	454,298.66
10	RandomForestRegressor	93.79	87.90	9.80	13.42	350,777.32	483,407.64
4	Gradient Boosting Regressor	86.72	85.03	13.62	14.57	513,048.82	537,649.30
3	DecisionTreesRegressor	99.84	84.50	0.23	10.52	56,363.16	547,212.05
5	K Nearest Neighbours Regressor	99.84	84.19	0.23	11.60	56,983.67	552,620.97
12	Support Vector Machines Regressor	80.27	77.35	15.87	17.72	625,419.62	661,497.12
6	Lasso Regression	75.15	74.34	20.98	20.98	701,890.38	703,993.84
11	RidgeRegressor	75.16	74.32	20.88	20.96	701,707.10	704,348.04
9	MultipleLinearRegression	75.17	74.30	20.85	20.97	701,660.45	704,610.79
14	ElasticNet Regression	73.49	73.63	22.20	21.75	724,939.81	713,656.48
1	AdaBoost Regressor	72.86	72.03	22.56	22.53	733,530.31	735,074.02
7	MLP Regressor	48.96	48.04	31.52	31.51	1,005,903.24	1,001,804.14
13	LinearSVR	-59.32	-60.65	60.23	59.92	1,777,185.95	1,761,560.67
8	SGDRegressor	-52,114,148,702,837.68	-47,200,749,697,672.50	6,892,406.59	6,126,466.12	1,016,427,455,113.17	954,830,568,327.06

Figure 3.2 Regressor comparison table

When contemplating on r2_train, r2_test, d_train, d_test, rmse_train, and rmse_test values shown in the above table, it is indisputable evidence that the triumphant models from the rest of the regressors are Extra trees regressor, Bagging regressor, Random Forest regressor, and Gradient Boosting regressor. Fundamentally, r2_test, d_test, and rmse_test were the criterion strongly considered since the training set may have led to overfitting, and those values could be misleading. Successful four regressor models with hyperparameter values (considering defaults) are shown below in Table 3.1.

Table 3.1 Top 4 regressor model optimal hyperparameters

Bagging	***Extra Trees***	***Random Forest***	***Gradient Boosting***
base_estimator=None, bootstrap=True, bootstrap_features=False, max_features=1.0, max_samples=1.0, n_estimators=10, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False	bootstrap=False, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf= 0, n_estimators=100, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False	bootstrap=True, criterion='mse', max_depth=30, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=360, n_jobs=None, oob_score=False, random_state=4, verbose=0, warm_start=False	alpha=0.9, criterion='friedman_mse', init=None, learning_rate=0.05, loss='huber', max_depth=4, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=15, min_samples_split=10, min_weight_fraction_leaf=0.0, n_estimators=15000, n_iter_no_change=None, presort='auto', random_state=None, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False

The models were further considered and analyzed in order to achieve the next crucial milestone of this research component, the stacking generalization ensemble regressor creation. Four models were chosen since that number of regressors for an ensemble model creation is stabilized and neutral. If it were to be more than 4, it would then account for all those prediction accuracies and error values, and the final ensemble prediction would start to vary in an inaccurate way due to a higher number of prediction opinions received. If less than 4 models were taken, a more generalized overview wouldn’t be provided, and the sole purpose of ensemble learning would be discarded. Ensemble learning explicitly focuses on improving the predictive performance and computational intelligence of the regressors by combining and harnessing those excellently performing models into one single hybrid model. Even in reality, if there is a single person and a group of people, the group would perform better in brainstorming sessions, and numerous idea or solution generation will be prompted for problems, ensuring a significant variety in opinions as well. Hence, the same goes

for the ML models, and the need for an adequate amount of regressors is apparent in order for the ensemble model to perform with an enhanced prediction power. In the comparison table, the R^2 value gave an implication of the goodness of regressor model fit or the variance ratio of the price values with respect to its independent estimators. However, this research decided to further analyze the R^2 value's validity of the final selected four models, and for that mission, the modified version of R^2, which is adjusted R^2 metric interpretation, was preferred and exploited. Thereby, a more precise view of the independent metric correlation can be provided and can certify the predictor suitability and variability as well. Below (6) is the equation of adj. R^2.

$$\text{Adjusted R} - \text{squared} = \quad 1 - \frac{(1 - R^2)(N - 1)}{N - P - 1} \qquad (6)$$

The python code used to calculate the adjusted R-squared value is shown below in Fig. 3.3.

```
In [24]: #display adjusted R-squared
         1 - (1-model.score(X, y))*(len(y)-1)/(len(y)-X.shape[1]-1)
```

Figure 3.3 Adjusted R-squared value calculation code

The adjusted R-squared values of the outstanding regressors are listed below, indicating a better model fit.

Bagging Regressor – 0.956

Extra Trees Regressor – 0.972

Random Forest Regressor – 0.920

Gradient Boosting Regressor - 0.872

By interpreting the above values, which all above 0.87 (from a scale of 0 to 1 for adj. R^2), it is undoubtedly clear that the models are certainly performing with high accuracy and prediction strength, making the choice of them for the ensemble stacked regressor authentic. In addition to that, the predicted price and the actual price dispersed graphs of the four regressor models and the best fit line is visualized below by Fig. 3.4.

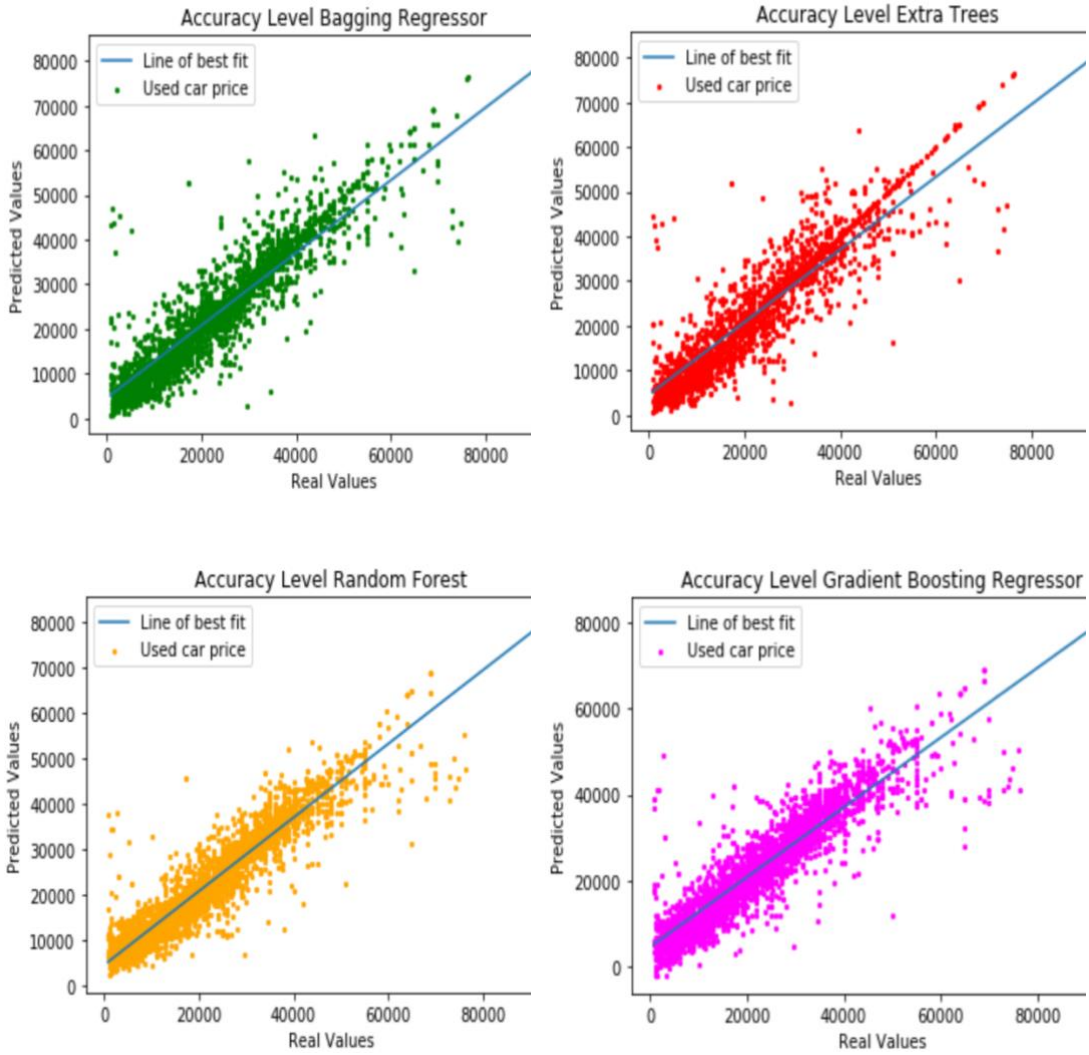


Figure 3.4 Top 4 regressor predicted values vs. actual values

When examining the above graphs, it is again obvious that the models have performed well when predicting the price value in comparison to the actual value since the majority of the dots in the four graphs are condensed together, where the predicted and actual value representing dots are overlapping on one another while forming the linear progressing path. Only a very few dots are perceptible as scattered though it is extremely negligible in the current circumstanced having a complex dataset.

Moving on, the stacked regressor for an ensemble generalization was created using the above eloquent models, and its efficiency was grandly noticeable in comparison to the solitary four models discussed above. For the base learners (level_0), Bagging, Extra Trees, and Random Forest regressors were used since those base-level learners should

have the premium performance out of all four. Hence, the Gradient Boosting regressor having the marginally low prediction power was (adj. R^2 0.872) chosen as the meta-learner (level_1) for the stacking model.

Following Fig. 3.5 shows the difference between the mean and standard deviation values of the four models' and the stacked regressor model's prediction scores. The python code written to obtain the result is depicted as well. The stacked regressor model has performed exceptionally well compared to the other four by producing -1850.050 as the mean score value and 103.634 as the standard deviation (std.) score. The lower the minus value, the better when it comes to analyzing the below mean score statistics and the std scores having a higher positive value is preferable.

```
In [25]:
from sklearn.model_selection import RepeatedKFold
from numpy import mean
from numpy import std
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
```

```
>bagging -2220.329 (82.770)
>extraTree -1920.138 (102.622)
>randomForest -3090.368 (90.082)
>gradientBoosting -3161.644 (98.659)
>stacking -1850.050 (103.634)
```

Figure 3.5 Individual and ensemble model accuracy comparison

Additionally, the below Fig. 3.6 provides a perspicuous view of the above mean score values graphically by using a boxplot to illustrate the marginal difference between the price predictions scores of all four models individually and the ensemble stacked regressor. Probing the graph itself, the ensemble model's (stacking's) enhanced prediction power is boldly distinguishable.

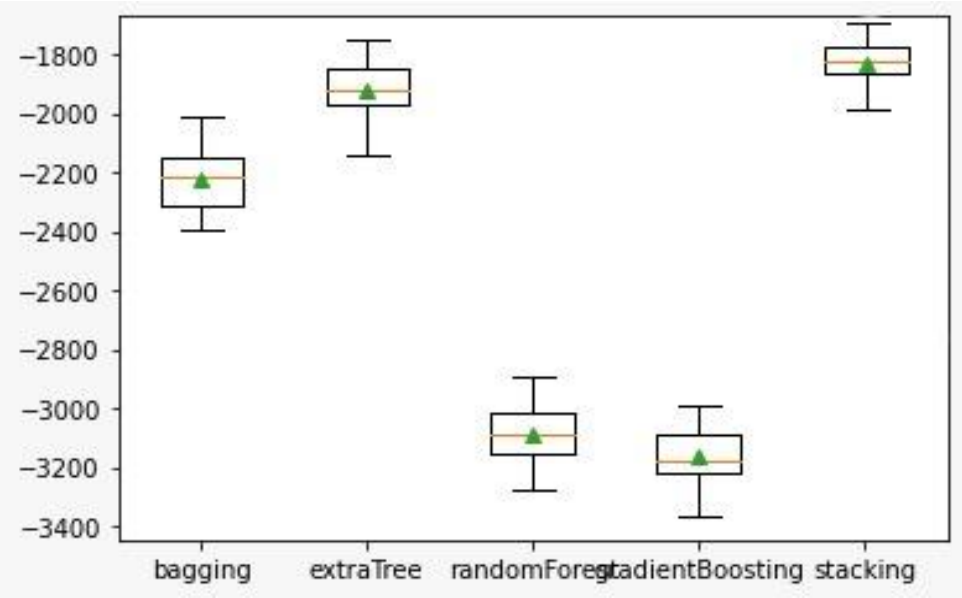


Figure 3.6 Individual and ensemble model mean score boxplot comparison

Due to the overly satisfying results received from the stacked regressor ensemble model, it was then serialized, and again deserialization with the API implemented using Flask framework, for it is utilized in the future by the client-side application when the user prompts the price prediction of a specific car being sold as a classified advertisement. All API implementation and testing details were stated in the previous chapter 'Methodology.' After both model and API integration was done, it was deployed into the cloud server environment, and a specific domain Uniform Resource Locator (URL) was assigned to the API endpoint as well for convenient accessing by the front-end Tievs application.

The front-end Tievs application was subsequently developed, and a quick glance of its homepage view can be taken through Fig. 3.7 below, which can be accessed using the URL 'www.tievs.com.' The price prediction facility served page is depicted by Fig. 3.8, which is a stepper page in the advertisement submission form, where the user enters all the essential characteristics of the vehicle being sold, which in turn facilitates the ML ensemble model's price prediction execution by invoking the API call from the client-side application, and finally the predicted price visualization. Thus, these results culminate in the substantial breakthroughs of this research.

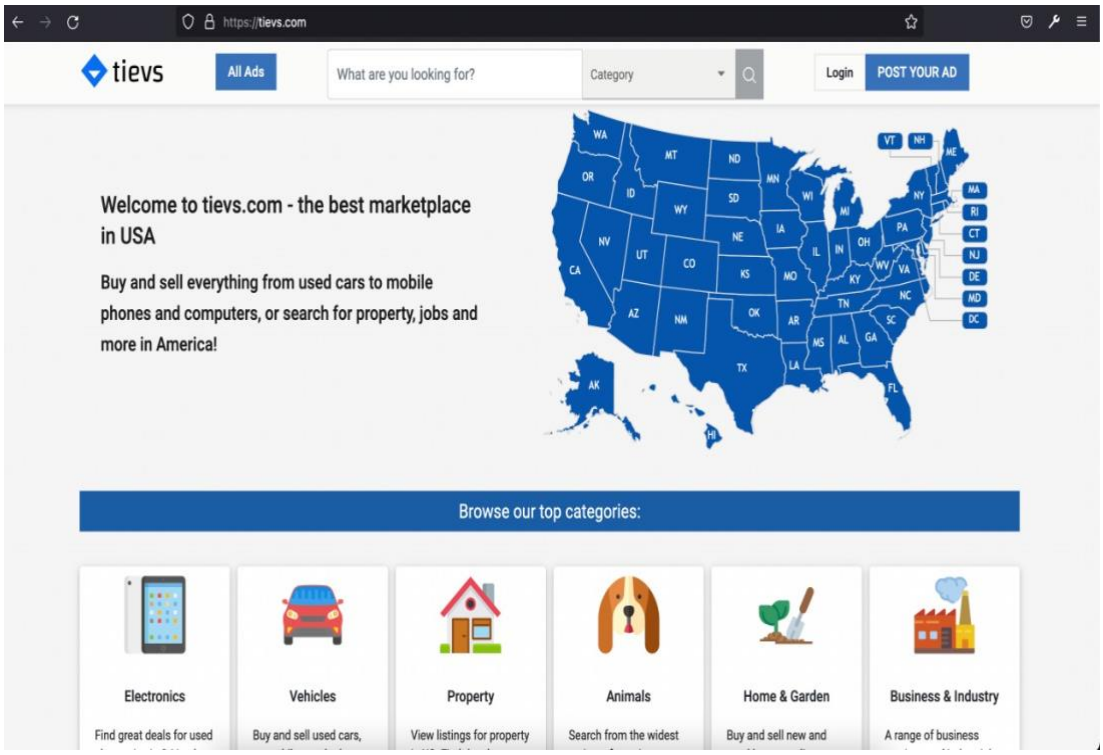


Figure 3.7 Front-end application home page

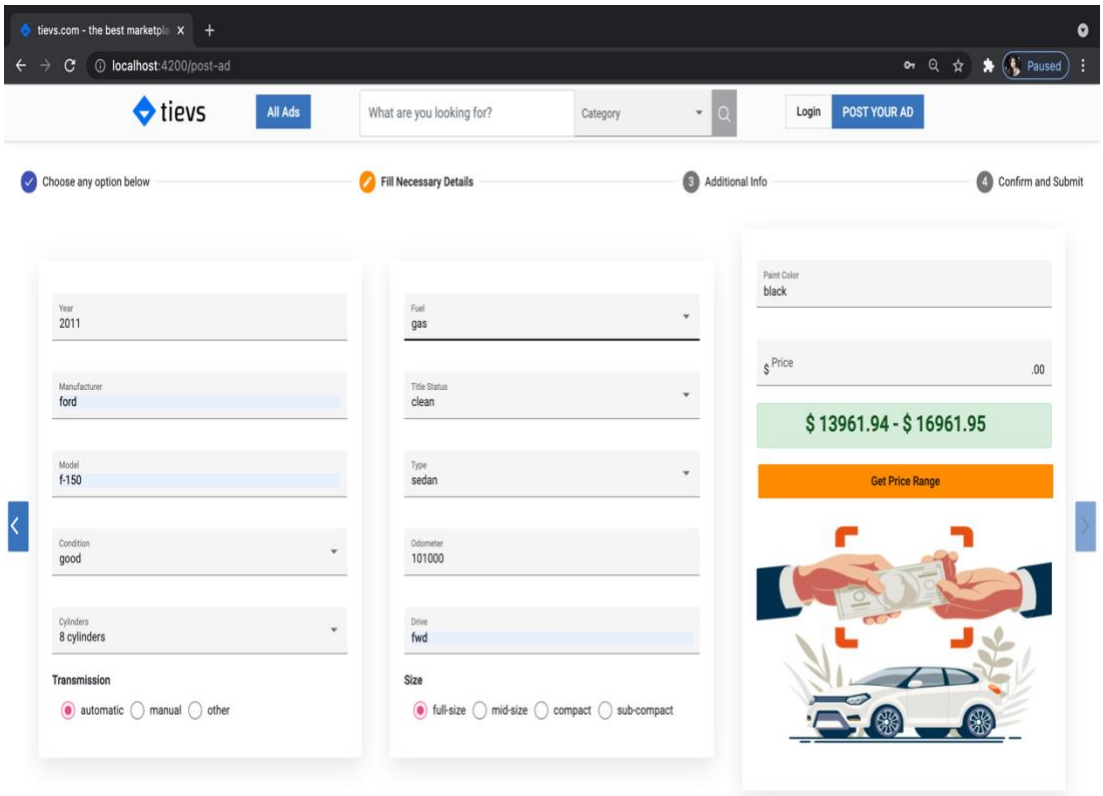


Figure 3.8 Predicted price visualization in ad submission form

The below shown Fig. 3.9 and Fig. 3.10 demonstrate the graphs and other exceptional performance-related metrics resulted from the wholly integrated Tievs classified adverting application, with all four major components, and especially the price prediction API and model amalgamated. The internet-found tool used for this analysis is GTmetrix, and it shows the performance, speed, web vitals, and other top issues.

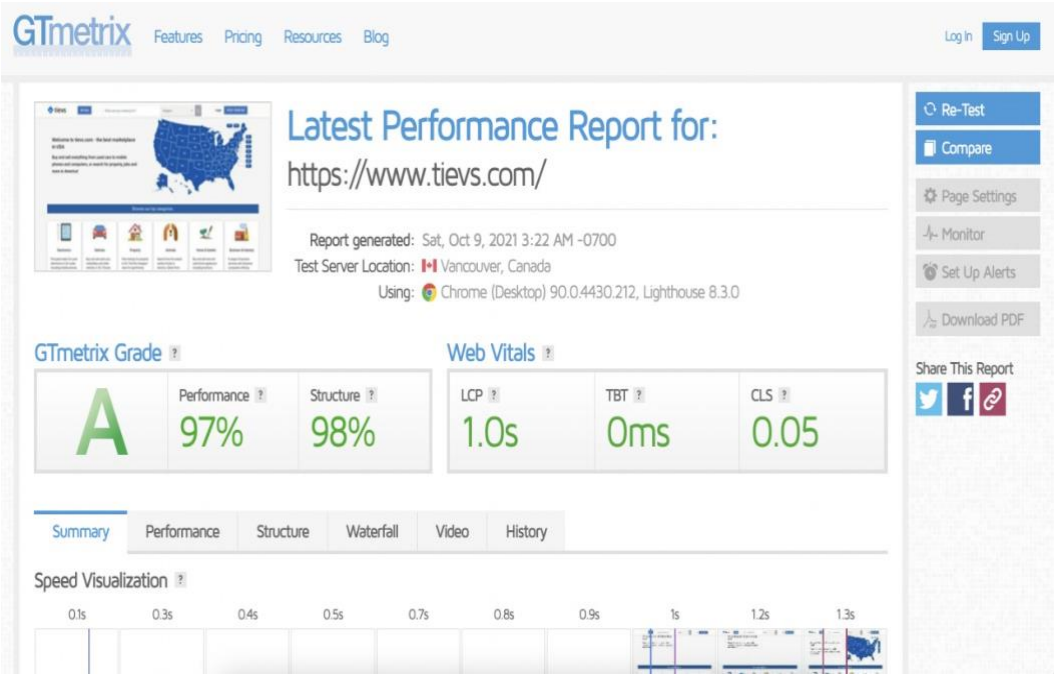


Figure 3.9 Tievs efficiency report from GTmetrix part 1

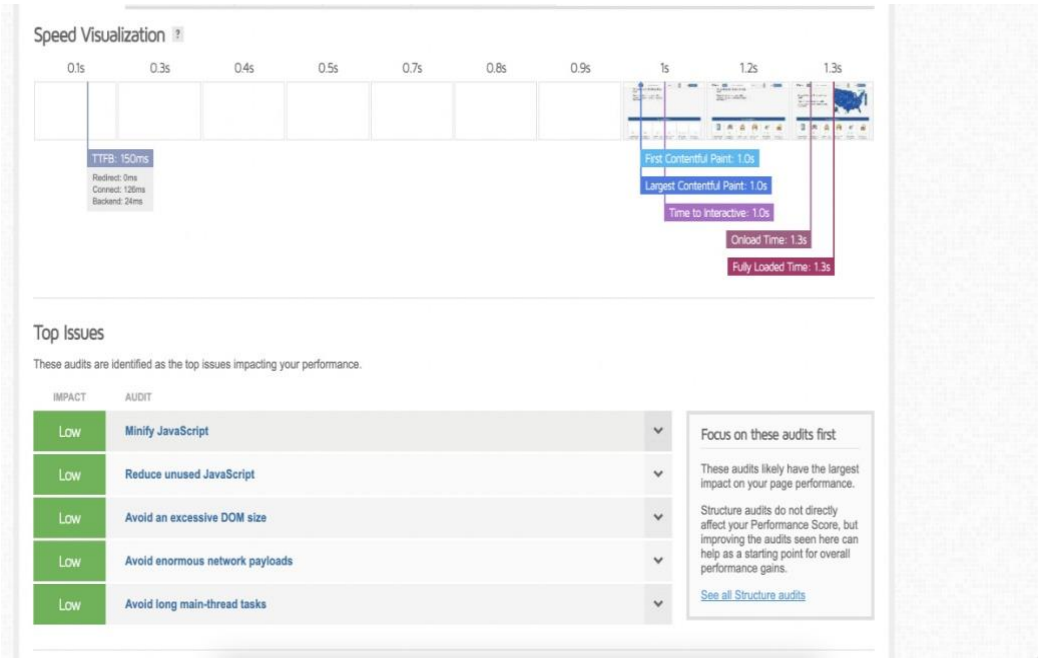


Figure 3.10 Tievs efficiency report from GTmetrix part 2

3.2 Research Findings

A synopsis for this section of the dissertation was constructed after the articulation of major research results pertaining to the price prediction system implementation regarding the enhanced classified advertising system depth into the key findings obtained by this research execution from the very start to the end.

Fundamentally, at the start of this study, when scrutinizing the research area and topic, it was found that not many previous analysis concluded have kept their focus towards the domain of classified advertising. E-commerce web application improvements, refinements, and even entirely novel components have been discussed, all the while avoiding the online classified advertising platform aspect, regardless of their soaring popularity nowadays, especially during a pandemic situation.

Moreover, research may have been done on used car price prediction [5]-[15], although none of those studies have incorporated the mechanism into the classified advertising application services to offer the end customers some comfort when trying to clarify a price value for the item they are hoping to sell. Existing literature objectified mainly on what kind of ML algorithm would perform in an outstanding manner with regard to price prediction, and after an outcome was attained, the experimentations concluded without proceeding further. Hence this research undertook the challenge to not only explore and investigate the most suitable ML model implementation but also to integrate it with a rich, user-friendly front-end application so that the functionality and final prediction visualization would be a real-world service to a prospective customer.

Furthermore, it was encountered that utilizing a classified application's own ML-based procedure for the evaluation of a price value for a particular selling item was quite advantageous for the application's internal functionality and the system developers' expediency. Though some existing classified platforms utilize website crawlers or similar third party appliances to provide their customers the price evaluation of an item they are selling, it is troublesome due to ambivalent network traffic originating to and

from the servers providing the price evaluation service, sudden server failures, incompatible or rigid attributes under consideration and future system augmentation immobilization. Developers won't have the intrinsic authority and control of the price evaluation procedures, and if faults were to occur from the dependent third parties, the application would lose price evaluation functionality altogether, making it incapacitated and risk losing reputation among consumers as well.

Out of all of the independent estimators of the price value, the make year and the mileage value were found the most influential from the study of the dataset. The other 11 attributes considered (mentioned in the 'Methodology') had only an adequate impact on price though those weren't omitted completely from applying into the prediction model. In addition to that, it was found that when using the K-fold cross-validation (CV) method, the model tendency to overfit was significantly decreased to a lower level. Training and testing the models with a different number of folds and 'K' times lead to analyzing the ML models' ability to predict unseen or new data. Due to that, the models were not overly or exactly matching against the training dataset, and they were performing poorly when it came to the test dataset. Without using the cross-validation method, the training accuracy seemed excessively large, indicating overfitting error.

However, after the implication of the K-fold CV method, the accuracy rates slightly decreased, showcasing the genuine metrics of the model, although the generalization aspect bolstered drastically. Thus, this research gave priority to predict prices of unseen second-hand car details without tolerating model overfitting, at the expense of faint presented preciseness subside. Nonetheless, RandomizedSearchCV and GridSearchCV assisted in finalizing the optimal hyperparameter values for the 15 regressor models that were examined, which in return addressed the accuracy declined issue and boosted up the perfection level of the models to an exemplary level yet again.

The regression-based supervised machine learning method was found highly appropriate for this research since the dependent variable is a continuous value to be predicted. When contemplating on the results obtained, it was conceivable that most

of the selected 15 regressors models and even all of the finally chosen winning models were all of decision tree type algorithms rather than being linear regression or neural network type. By analyzing the comparison table with the final values of 15 regressor performance criteria, it is apparent that decision trees led to better prediction capabilities. It was found that the intuitive performance and the fewer data preprocessing necessity, along with not requiring scaling or normalization of values to a significant extent, made those decision trees models function well withstanding minor insignificance and being harshly affected. However, excessive resource consumption and higher time for the training were identified as a considerable drawback as well.

This research incorporated ensemble learning under the rudiment hypothesis that it will eminently perform exceptionally well when comparing with individual algorithm models. For this purpose, stacked regressor generalization was utilized. It is usually tedious to encounter a model which performs best on a particular dataset. Hence, through an alternative combination of such good models, a sublime performance was established, which outperforms solidarity models. All in all, the hypothesis was ultimately proven and found true when examining the results generated from this research regarding the ensemble model.

Finally, when implementing the API, at first, the basic Flask framework was used, although going forward, it was found that in order to facilitate Hypertext Transfer Protocol Secure (HTTPS) and CORS, the Quart framework was necessary and was applied. It was an unexpected finding to know that Flask alone could not effectuate the HTTP/2 protocol transmission permission.

Nevertheless, it is safe to say that the conducted research achieved noteworthy findings that contribute novel perspectives to its relative technological and business domain while ameliorating classified advertising portals to serve its consumers admirably, in relevance to price prediction.

3.3 Discussion

This research was fundamentally conceived to produce an application-independent price prediction mechanism to provide the customers with a reliable and convenient price value when they submit a new classified advertisement into the advertisement portal. Due to the scope and time frame provided for this research, out of all classified types, the 'Car' classifieds were chosen and exploited for development.

Although most prevailing systems have utilized website crawlers to obtain a price evaluation for a specific item of interest, this research analyzed a different path where the prediction system may reside within the control and authority of the application developer. Thereby, any unnecessary third-party dependencies and networking and traffic-wise obstacles for the classified application were found eliminated by the end of this experimentation. Moreover, most classified advertising portals have not exercised sophisticated technologies such as ML for their application production deployment. Thus this research constructed the hypothesis of its' feasibility, which was proven true as resolution since that objective was achieved in the end.

The utilization of an openly available dataset promoted the quick implementation of the research's core aims and objectives without misspending time unnecessarily since the dataset was more than required for a price prediction methodology such as in this study [5]-[10]. It was an openly available dataset called 'Used cars dataset,' and it had 441,396 second-hand car advertisement records, which were scraped from Craiglist. During the data processing and exploratory analysis phase, the complexity of the dataset was thoroughly recognized. Regardless, this research proceeded to consider all 441,396 records for model training and testing. Other studies [5]-[15] have been reluctant to utilize such a complex dataset for their experimentations since the data engineering part requires meticulous handling, which this research bravely tackled. However, due to insufficient computational resources and the dataset size being more than 1.4 GB, the memory and Central Processing Unit (CPU) capacity overloaded and Jupyter Notebook kernel crashed. The below Fig. 3.11 shows the specifications of the machine which maneuvered this whole research.

Figure 3.11 Research-Conducted Machine Overview

Even the simplest ML code executions did not seem to run without interruptions under the current circumstances. Therefore, the scope of the dataset had to be reduced with respect to the manufacturer and model of the dataset, which led to selecting only Ford, Toyota, and Chevrolet vehicles of the F-150 model type. Then, the remnant dataset consisted of 14,956 records, which was still higher than what the studies [5]-[15] procured. However, by doing that, noise and outliers were eliminated, which declined overfitting, making the models have appropriate bias and variance levels.

It was skeptical initially regarding the attainability of the comparative study of 15 different regressors, which seemed quite challenging when considering this research's scope. However, the goal was accomplished by the smooth but rigorous execution of the procedures. Previous studies [5]-[15] had done similar comparative studies on multiple ML algorithms, although they were limited to a smaller number, not exceeding 6. In that sense, this research advances further in the strategy undertaken.

The regressors considered in this study were Extra Trees, Bagging, RF, Gradient Boosting, Decision Trees, K-nearest Neighbors (KNN), Support Vector Machine (SVM), Lasso, Ridge, Elastic Net, AdaBoost, Multilayer Perceptron (MLP), Linear SVR, Stochastic Gradient Descent (SGD), and Multiple Linear Regressor. They were trained and tested using the same dataset in a 75:25 split ratio with 10 K-folds for cross-validation, which in turn helped to reduce the overfitting problem and promote

generalization. GridSearchCV and GridSearchCV mechanisms were used to tune the hyperparameters of the above models and gain the optimum values for an accurate prediction. The majority of the 15 regressors consisted of decision trees due to their effectiveness against prediction, dataset manipulation, and model fitting ability. Still, it confiscated a lengthy time period for regressor training.

Another rationale on why this research is significant and innovative when comparing with similar existing research is that most of them have only evaluated the algorithm prediction performances by analyzing them individually with the relevant hyperparameters. Ensemble learning is rarely implored, especially in car price prediction [5]-[14]. The study [15], being divergent, employed only a classification-based ensemble model. Hence, investigations on the regressors' capacity to work as a group for prediction are further asserted obscure. Contrastingly, this research expanded the regressor ensemble research area by filling those gaps.

Out of all 15 models, Extra Trees, Bagging, RF, and Gradient Boosting regressors were deemed the most exemplary, having higher R^2, adj. R^2 values (above 0.87 from the scale of 0 to 1) and lower MAE, MSE, and RMSE values from the test dataset. Since ensemble learning was appointed as one of the main targets of this research, the above four regressors were combined together using a heterogeneous ensemble learning technique called stacked generalization.

To build the stacking regressor, Extra Trees, Bagging, RF were used as the base learners since those three were at the top from all four, and Gradient Boosting regressor was chosen as the meta-learner, and it gave the final predictions according to the prediction results obtained from the base learners. Although the top four regressors were commendable when performing individually, it was found while proving the initial hypothesis at the beginning of this research, the stacked or the ensemble model performed even more exceptional when predicting the price value according to the car features provided. This hybrid prediction power having commendable generalization, as depicted in the 'Results' subsection, unquestionably escalates novelty and is a huge find while being an asset to this research’s results.

Studies [5]-[14] have concluded implying the best model, although the experimentation procedures ended from there onwards. The inclusion of the model into an API in order to deploy within a cloud environment so as to access it later on through a front-end application was not explored. Therefore, this research's methodology originality is again confirmed due to the front-end application development using a comprehensive framework such as Angular and the integration between the implemented API with it. The front-end system built with the Angular architecture and Typescript as the main programming language, which handled UI components, had its own back-end services, which were liable for invoking the appropriate API calls and requesting the model's forecasted results. Moreover, if the prediction model wasn't deployed in cloud storage, the service itself would be pointless due to inaccessibility by its other dependencies.

Python was chosen as the primary language to implement the API, and for that, Flask microframework was utilized due to its appropriacy in building APIs. However, later on, it was found that Flask itself could not accommodate HTTPS request calls through CORS headers. Hence, Quart, which is an extension of Flask, was confiscated, and the API source code structure was redesigned. Thus, the model residing with the API could extract car details from the UI advertisement form the customer fills, predict the accurate price value, and send the response with the final forecasted value into the client application for a user-friendly visualization. The API's POST HTTP verb method was used for this purpose with the endpoint of 'predict-price.'

In conclusion, it is distinctly evident by contemplating on the results and findings that this research was able to achieve applaudable outcomes and emphasize the need for an independent price prediction system within a classified advertising platform to address consumer necessities and provide solace when clarifying a necessary price value for a certain product, which in this scenario, 'used cars.' The innovativeness of this research lies within the above-discussed eccentric units, which can proactively be enhanced in prospective aspirations and endeavors undertaken.

3.4 Summary of Student's Contribution

Table 3.2 Summary of personal contribution

Member	Component	Tasks
K.S.S. Bandaranayake	Used Car Ensemble Price Prediction Model	• Analyze existing literature to find out what has and has not been done with regard to attempts on used car price prediction. • Identify the most eligible ML algorithms that can perform well with regard to price prediction. • Accumulate an adequate dataset having all essential characteristics that will prompt a higher accuracy during the ML model implementation. • Perform necessary feature engineering and data preprocessing methods along with exploratory data analysis to get a better depth of the dataset being utilized. • Conduct a comparative study on 15 different algorithm models, train and test them using refined hyperparameter to obtain higher prediction accuracy and low error rates through specific performance metrics (R^2, adj. R^2, MAE, MSE, RMSE, RSE). • Selection of the top four exemplary models out of all 15 models through metric analyzation and creation of an ensemble model following the stacked generalization architecture for a powerful combined prediction. • Implement the ML API to allow the ensemble model to be integrated and utilized through an HTTP POST endpoint and deployment of the price prediction API in a cloud environment. • Front-end application necessary component design and development. • Integration of price prediction API and Tievs front-end application's advertisement submission form to accept predict price request and final predicted price range visualization.

4 CONCLUSION

The fundamental objective of this dissertation-based research was to design and implement a dynamic price prediction mechanism, using an ML model, to recommend or suggest a convenient and reliable price value for a specific product a customer is trying to sell through a classified advertising system. Due to the scope of this research, designing the system to accommodate all types of classifieds was not practical or feasible. Hence, 'Car' type classifieds were chosen foremostly to continue with the rest of the procedures. The popularity of classified advertising platforms has been soaring high in recent years, and its predecessor periodicals, TV or radio channel advertising has been lees favored and losing their fame day by day.

Since the accessibility is easy without spending any money if the internet is being provided, classified advertising is preferred by all around the world for quick views of items on sale, regardless if they are brand new or used ones. The availability and user-friendly interfaces play a major role as well to uplift the commercialized favoritism. Some of the infamous classified advertising applications are Craiglist, Olx, Quikr, Carwale, CarDheko, ikman.lk, and Facebook Marketplace. This inevitability of online classified advertising, along with rapid technological growth and digitalization, has made it quite lucrative for both sellers and shoppers in their own endeavors. On the other hand, the number of used cars within the market has been expeditiously increasingly due to various reasons such as fewer funds available at the disposal of buyers, vehicle leasing scheme, overpriced brand new cars, and the exaggerating number of people who tends to change vehicles within a short time period.

As a result, it is quite tedious, especially for car resellers, to estimate a realistic price value for their vehicle when they put it up on sale. These are mostly done through online classified platforms nowadays, and when customers of those applications try to submit an ad about their reselling car, they waste valuable time out of their busy schedules to do exhaustive research themselves to clarify a certain price value. Hence, the need arises for a price prediction system to be incorporated within such a classified advertising system to facilitate the customers with a fair price value then and there

itself, without allowing them to wander off to another website or portal in search of the exact prices. Then a research question was inaugurated to indicate the necessity of such a prediction system since some prevailing systems utilize website crawlers to acquire the relevant price values for their customers. However, that is definitely not the best way to manage the issue at hand since web crawlers are third-party applications, and it is doubtful if they will face server/networking failures due to high traffic or internal problems. The specific attributes that they consider might be different from our own classified application concerning attributes, and scaling of those would not be possible as well. Any version dependencies would need to be accounted for, and classified ad application immobilization is inevitable. The developer of the application would have complete authority and control over the mechanism if it is to be an independent, application-based price prediction system.

ML techniques are the best when it comes to supervised learning prediction procedures. Both classification or regression type ML models can be found, although, for this study, regression was chosen since it can handle a continuous variable such as a price value more efficiently. The research was done in the past for implementing an ML model for the price prediction of used cars [5]-[15], which initially inspired this study, although some problems were present in their methodologies. Therefore, this research opted to thoroughly analyze past literature to identify the prevailing technical research problems and address them using the most appropriate solution. Most studies only had a limited number of data records and not complex datasets. Comparative studies that were conducted only considered about maximum 6 algorithms to reason out each of their prediction efficacy.

What’s more astounding is the scarce use of ensemble learning methodologies within the respective research domain. By using ensemble learning, the prediction power of multiple models can be harnessed and combined to produce a far superior forecasting strength. That functionalism was not explored up to a satisfactory level. Studies have only focused on building the ML model for prediction, and that was the conclusion of their whole experiment. Implementation of an API or a front-end application to provide a UI for prediction result visualization has been neglected and overlooked,

which in turn makes the constructed prediction models pointless since they would not be able to satisfy any consumer or business requirements. Even though those are research pertaining to ML model performance and accuracy evaluation, they still need to be utilized by the end consumers if they perform well in their own functionality in order to be more fruitful in existence. Addressing above mentioned problems was the main aim of this research component as a whole and to start the process, the 'Used Cars dataset' containing the record number decreased after data preprocessing was conducted to handle missing values, noise, and outliers. Furthermore, only the Toyota, Ford, and Chevrolet manufactured F-150 model vehicles were considered for model training and testing purposes due to limited computer resources. After all categorical variables were converted for numerical (0s or 1s) through OHE, the cleaned dataset was exported, and the next step of the comparative study on 15 different regressors started. Those models were repeatedly trained and then afterward tested using the most convenient hyperparameters. Cross-validation was used to eliminate the impact of overfitting to bolster generalization.

This research undertook this comparative study as a challenge itself and was able to accomplish satisfying results by the end where a sound knowledge and a deep understanding was originated regarding the above-stated ML regressors. In the end, Extra Trees Bagging, RF, and Gradient Boosting were chosen as the best models out of all 15 regressors and proceeded towards the stacking ensemble model creation using the stacked regressor, which further gave more outstanding prediction scores than any of those four models could provide singularly. This finding, which is thoroughly accentuated in the above 'Results & Discussion' chapter, will hopefully make its stand among other major studies and advocate to disseminate the ensemble learning techniques even further so that it will be utilized regarding the development of hybrid models that are robust and precise in prediction functionality. The next major phase that commenced after the successful implementation of the ensemble model was the API and front-end application development. The model was integrated with the Quart framework implemented API's POST HTTP method and 'predict-price' endpoint to provide its prediction results through a client-side application, which would be more pleasing and suitable, especially since its main task is to serve classified advertisement

submission by forecasting an expedient price value for the customers. The Angular framework-built front-end application had its own services as a back-end, which was responsible for invoking the necessary API calls and requests for the prediction results out of the model. In the end, both the application and the model included API with other dependencies were deployed in a cloud droplet for higher availability and accessibility.

Nonetheless, space for future enhancements is present in this research component. Supplemental tuning of the model hyperparameters could be enacted to reduce the RMSE and RSE values considerably and increase prediction accuracy (R^2 and adj. R^2) to be above 90% (especially of Gradient Boosting regressor). As another future work, experimenting on more sophisticated algorithms as genetic, fuzzy logic, reinforcement learning, and neural networks could also be prosecuted to verify how those models are able to predict the prices as opposed to this research's comparative study included regressors. In addition, the accommodation of other types of classified will be considered as a future work of the overall Tievs application. Due to that, the price prediction for those types of classifieds such as real estate, electronic devices, pets, cosmetic items, food items, etc., could and should be implemented as well. For that, the necessary datasets and the optimum models for prediction would need to be analyzed from the get-go processes and certainly would be a tremendously taxing mission altogether, which needs conscientious attentiveness to each step within the methodology designed and followed. Unification of online model retraining is a captivating arrangement for this research component; since then, it would be possible to keep the model updated with the new incoming car classified data, as the price is an attribute that tends to deviate in accordance with changing market trends. The model will then adapt and learn sequentially by accounting for all recent observations to enhance its prediction power. Hence, if this practicality is adopted, a complex dataset won't be an issue since training all records at once is not necessarily attempted.

All things considered, the price prediction model and the Tievs web application should utterly be polished along with other neighboring key components prior to being released into production and, consequently, to customers.

REFERENCES

[1] NATIONAL TRANSPORT AUTHORITY. 2014. Available from: http://nta.gov.mu/English/Statistics/Pages/Archive.aspx [Accessed Feb. 19, 2021].

[2] WOLF STREET. (2019). "Average Age of Vehicles Sets Record, New-Vehicle Sales Drop to Where They Were 20 Years Ago. What Are Automakers Doing?," [online] Available: https://wolfstreet.com/2019/06/27/good-for-consumers-part-of-carmageddon-for-automakers-average-age-vehicles-in-operation/ [Accessed Feb. 19, 2021].

[3] ACCESS. (2012). "Will China's Vehicle Population Grow Even Faster than Forecasted?," [Online] Available: https://www.accessmagazine.org/fall-2012/will-chinas-vehicle-population-grow-even-faster-forecasted/ [Accessed Feb. 19, 2021].

[4] GREEN CAR CONGRESS. (2019). "US used vehicle sales are more than double the number of new vehicle sales," [Online] Available: https://www.greencarcongress.com/2019/07/20190716-fotw.html [Accessed Feb. 19, 2021].

[5] Ganesh, Mukkesh & Venkatasubbu, Pattabiraman, "Used Cars Price Prediction using Supervised Learning Techniques," *International Journal of Engineering and Advanced Technology*, vol. 9, Dec., pp. 216-223, 2019.

[6] Kuiper, Shonda. "Introduction to Multiple Regression: How Much Is Your CarWorth?," *Journal of Statistics Education 16.3*, 2008.

[7] N. Kanwal and J. Sadaqat, "Vehicle Price Prediction System using Machine Learning Techniques," *International Jounal of Computer Applications*, vol. 167, no. 9, pp. 27–31, 2017.

[8] N. Monburinon, P. Chertchom, T. Kaewkiriya, S. Rungpheung, S. Buya and P. Boonpou, "Prediction of prices for used car by using regression models," 2018 5th International Conference on Business and Industrial Research (ICBIR), 2018, pp. 115-119, doi: 10.1109/ICBIR.2018.8391177.

[9] Xinyuan Zhang , Zhiye Zhang and Changtong Qiu, “Model of Predicting the Price Range of Used Car,” 2017.

[10] N. Pal, P. Arora, S. Sumanth Palakurthy, D. Sundararaman, P Kholi, “ How much is my car worth? A methodology for predicting used cars prices using Random Forest,” Future of Information and Communications Conference (FICC*)*, 2018, pp. 1-6.

[11] S. Pudaruth “Predicting the Price of Used Cars Using Machine Learning Techniques,” *International Journal of information & Computation Technology*, vol. 4, no. 7, pp. 753-764, 2014.

[12] S. Peerun, N. H. Chummun, and S. Pudaruth, “Predicting the Price of Second-hand Cars using Artificial Neural Networks,” The Second International Conference on Data Mining, Internet Computing, and Big Data, 2015, pp. 17–21.

[13] N.Sun, H. Bai, Y. Geng, and H. Shi, “Price Evaluation Model in Second-hand Car System based on BP Neural Network Theory,” 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017, pp. 431–436.

[14] M. Listiani, “Support Vector Regression Analysis for Price Prediction in a Car Leasing Application,” Master Thesis, Information and Media Technology, Hamburg University of Technology, pp. 1-85, 2009.

[15] E. Gegic, B. Isakovic, D. Keco, Z. Masetic, J. Kevric, “Car Price Prediction using Machine Learning Techniques,” International Burch University, Sarajevo, Bosnia and Herzegovina, vol. 8, no. 1, pp. 113-118, 2015.

[16] G.Chandrashekar and F. Sahin, “A survey on featureselection methods,” Computers & Electrical Engineering, vol. 40, no. 1, pp. 16–28, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790613003066 [Accessed Feb. 22, 2021].

[17] R.Taylor, “Interpretation of the Correlation Coefficient: A Basic Re- view,” *Journal of Diagnostic Medical Sonography*, vol. 6, no. 1, pp. 35–39, 1990.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander- (5)plas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duches-nay, “Scikit-learn: Machine Learning in fPgython,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] P. Geurts, D. Ernst, L. Wehenkel, “Extremely randomized trees,” *Mach Learn*, vol. 63, pp. 3-42, March, 2006. [Online serial]. Available: https://link.springer.com/content/pdf/10.1007/s10994-006-6226-1.pdf [Accessed Aug. 2, 2021].

[20] L. Breiman, “Bagging predictors,” *Mach Learn*, vol. 24, Aug., pp. 123-140, 1996.

[21] Tin Kam Ho, "Random decision forests," Proceedings of 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, vol.1, 1995, pp. 278-282.

[22] L. Mason, J. Baxter, P. Bartlett and M. Frean, “Boosting Algorithms as Gradient Descent,” NIPS'99: Proceedings of the 12th International Conference on Neural Information Processing Systems, 1999, pp. 512-518.

[23] Dr. Saed Sayad, “Decision-Tree Regression,” July, 2021. [Online]. Available: https://www.saedsayad.com/decision_tree_reg.htm [Accessed Aug. 14, 2021].

[24] Fionn Murtagh, “Multilayer perceptrons for classification and regression,” *Neurocomputing*, vol. 2, no. 5-6, pp. 183-197, 1991.

[25] J. Brownlee, “Boosting and AdaBoost for Machine Learning,” *Machine Learning Mastery*, April. 25, 2016. [Online]. Available: https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/ [Accessed Aug. 15, 2021].

[26] N. S. Altman, “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression,” *The American Statistician*, vol. 46, no. 3, Aug., pp. 175-185, 1992.

[27] E. Hoerl and Robert W. Kennard, “Ridge Regression: Biased Estimation for Nonorthogonal Problems,” *Technometrics*, vol. 42, no. 1, Feb., pp. 80–86, 2000.

[28] A. Hayes, “Multiple Linear Regression (MLR),” *Investopedia*, March. 30, 2021. [Online]. Available: https://www.investopedia.com/terms/m/mlr.asp [Accessed Aug. 16, 2021].

[29] R. Tibshirani, "Regression Shrinkage and Selection via the lasso," *Journal of the Royal Statistical Society. Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[30] H. Zou and T. Hastie, “Regularization and variable selection via the Elastic Net,” *Journal of the Royal Statistical Society, Series B*, vol. 67, pp. 301-320, 2005.

[31] Jerome H. Friedman, “Stochastic Gradient Boosting,” *Computational Statistics & Data Analysis*, vol. 38, no. 4, Feb., pp. 367-378, 2002.

[32] A. Ben-Hur, D. Horn, H.T. Siegelmann, V. Vapnik, “Support Vector Clustering,” *Journal of Machine Learning Research*, vol. 2, Nov., pp. 125-137, 2001.

[33] J. A. K. Suykens, J. Vandewalle, “Least Squares Support Vector Machine Classifiers,” *Neural Processing Letters*, vol. 9, no. 3, Jun., pp. 293–300, 1999.

[34] Necati Demir, “Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Products,” toptal.com, 2016. [Online]. Available: https://www.toptal.com/machine-learning/ensemble-methods-machine-learning [Accessed Aug. 20, 2021].

[35] Rosaria Silipo, "Ensemble Models: Bagging & Boosting," *Analytics Vidhya*, March. 19, 2020. [Online]. Available: https://medium.com/analytics-vidhya/ensemble-models-bagging-boosting-c33706db0b0b [Accessed Aug. 21, 2021].

[36] KDnuggets, "Ensemble Learning to Improve Machine Learning Results - KDnuggets," *KDnuggets*, 2021. [Online]. Available: https://www.kdnuggets.com/2017/09/ensemble-learning-improve-machine-learning-results.html/2 [Accessed Aug. 23, 2021].

APPENDICES

Appendix A: Price Prediction System Test Cases

Test Case ID	01
Test Case Description	Ensuring that all data values are successfully preprocessed.
Input	'vehicles.csv' dataset
Expected Output	All data values were successfully transformed and preprocessed.
Actual Output	All data values were successfully transformed and preprocessed.
Final Test Result	Pass

Test Case ID	02
Test Case Description	Ensuring that all column attributes relating to the price attribute are thoroughly understood through exploratory data analysis.
Input	'vehicles.csv' dataset
Expected Output	All price-related column values and relationships are visually realized.
Actual Output	All price-related column values and relationships are visually realized.
Final Test Result	Pass

Test Case ID	03
Test Case Description	Ensuring that all 15 regressors chosen have been efficiently trained.
Input	'cleanedData1.csv' dataset (split into train dataset), Scikit-Learn library imported 15 regressor models.
Expected Output	Satisfactory training accuracy and error rates received.
Actual Output	Satisfactory training accuracy and error rates received.
Final Test Result	Pass

Test Case ID	04
Test Case Description	Ensuring that all 15 regressors chosen have been efficiently tested.
Input	'cleanedData1.csv' dataset (split into test dataset), Scikit-Learn library imported 15 regressor models.
Expected Output	Satisfactory testing accuracy and error rates received.
Actual Output	Satisfactory testing accuracy and error rates received.
Final Test Result	Pass

Test Case ID	05
Test Case Description	Verification of the top four excellent regressor models.
Input	Extensively trained and tested 15 regressor models.
Expected Output	The top outstanding four regressors were distinctively identified.
Actual Output	The top outstanding four regressors were distinctively identified.
Final Test Result	Pass

Test Case ID	06
Test Case Description	The stability of the stacked ensemble model is verified through the combination of the four regressors chosen.
Input	'cleanedData1.csv' dataset (split into train/test), selected top four regressor models,
Expected Output	Receive satisfying accuracy and error criterion for the stacked ensemble model.
Actual Output	Satisfying accuracy and error criterion received for the stacked ensemble model.
Final Test Result	Pass

Test Case ID	07
Test Case Description	Efficient functioning implementation of Flask/Quart API verification.
Input	Deserialized ensemble model, model-execution related python dependencies.
Expected Output	Acquiring effectiveness of API functionality.
Actual Output	Effectiveness of API functionality received.
Final Test Result	Pass

Test Case ID	08
Test Case Description	Postman API testing and verification of model and API execution with final price prediction.
Input	JSON formatted car attributes for estimating price value, API server deployed URL
Expected Output	Accurate prediction of price value.
Actual Output	Accurate prediction of price value.
Final Test Result	Pass

Test Case ID	09
Test Case Description	Postman API testing and verification of the cloud deployed price prediction API server with the ensemble model.
Input	JSON formatted car attributes for estimating price value, cloud-deployed API server URL
Expected Output	Receive precise price prediction final result.
Actual Output	Precise price prediction final result received.
Final Test Result	Pass

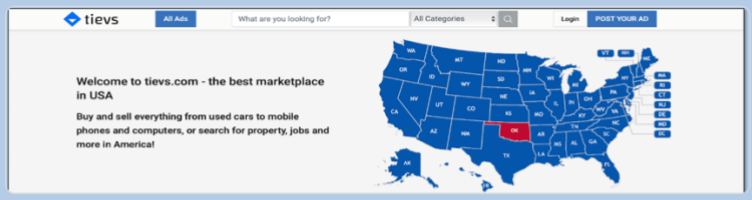
Test Case ID	10
Test Case Description	Testing if the front-end application's Angular framework implemented design component rendering happens without errors.
Input	'ng-serve' command through the VSCode terminal
Expected Output	Graphical user interface visualized without any errors.
Actual Output	Graphical user interface visualized without any errors.
Final Test Result	Pass

Test Case ID	11
Test Case Description	Testing the front-end application's Angular framework implemented design component functionality.
Input	Necessary car attributes, 'Get-price-range' button click
Expected Output	All components working as expected.
Actual Output	All components were working as expected.
Final Test Result	Pass

Test Case ID	12
Test Case Description	Ensuring advertisement submission form displays the API server responded final prediction result from the ensemble model.
Input	Necessary car attributes, 'Get-price-range' button click
Expected Output	The predicted price range being displayed accordingly.
Actual Output	The predicted price range was displayed accordingly.
Final Test Result	Pass

Appendix B: Tievs User Feedback Questionnaire

Questionnaire Link - https://forms.gle/TyCEopchFp7UsR4A7



"Tievs" Classified Advertising System's User Feedback Questionnaire

This questionnaire is used to gather customer feedback for the purpose of a SLIIT final year research project "Tievs: Classified Advertising Enhanced Using Machine Learning Techniques", which is being supervised by Ms. Manori Gamage and co-supervised by Ms. Suriyaa Kumari. The main expertise and reference person is Mrs. Wasanthi Bandaranayake.

Tievs is simply an ameliorated classified advertising platform that ventures on further to cater to the customer requirements of classified advertisement submission and exploration. The functionalities being provided by this system are being advanced through Machine Learning, Deep Learning, and Natural Language Processing concepts. The major components of Tievs are the optimised recommendation system, the independent classified advertised item price prediction system, fraudulent content detection, and prevention system, and finally erroneous image identification and avoidance system. Due to scope limitations, car classifieds are being effectuated for the time being although expansion for other classified types are underway.

Please visit the below URL to redirect to the Tievs domain where you are free to navigate through the application as you wish. Simultaneously, you can provide your valuable feedback by filling out this survey, with respect to your experience and perspective of Tievs classified advertising application.

https://tievs.com/

We ensure the data will remain anonymous as well as data will only be used for research purposes and will not be used for any commercial purpose.

By choosing to continue you will be giving your consent to participate in the study. Please confirm your consent below. *

☐ I give my consent to participate in this study

Please answer the following questions to the best of your knowledge.

1) Enter your age. *

Your answer

2) State your gender. *

- ○ Female
- ○ Male
- ○ Prefer not to say
- ○ Other: _____

3) From the following list, which are the applications that are more familiar with you with respect to classified advertising? *

- [ ] Craiglist
- [ ] Quikr
- [ ] Olx
- [ ] Carwale
- [ ] CarDheko
- [ ] Pakwheels
- [ ] ikman.lk
- [ ] KSL.com
- [ ] Oodle
- [ ] Hoobly
- [ ] Geebo
- [ ] Locanto
- [ ] FaceBook Marketplace

4) Which feature(s) are the most important to you in a classified advertising system? *

- [ ] Convenient and appropriate classified ad recommendation
- [ ] Price prediction and visualisation for an item prior for it to be advertised
- [ ] Fraud content identification and prevention from being submitted into the portal
- [ ] Fraud image detection and elimination to protect platform reliability and authenticity

5) Do you find the graphical user interface of Tievs attractive? *

- ( ) Yes
- ( ) No
- ( ) A little
- ( ) Not much

6) Do you think it is easy to navigate through the sections within the Tievs web application? *

- Yes
- No
- Maybe
- I'm not sure

7) Did you encounter any inconveniences while browsing or utilising the functionalities provided by Tievs? *

- Yes
- No
- Maybe

8) If the answer you gave is "Yes" for the previous question (7), please state the nature of the inconvenience which you faced. (Else, you can ignore this question)

Your answer

9) How do you assess the effectiveness of the recommendation system in Tievs? *

- Very Good
- Good
- Poor

10) How do you assess the effectiveness of the car price prediction and visualisation system in Tievs? *

- Very Good
- Good
- Poor

11) How do you assess the effectiveness of the fraud content identification and prevention system in Tievs? *

- Very Good
- Good
- Poor

12) How do you assess the effectiveness of the erroneous image identification and prevention system in Tievs? *

- ◯ Very Good
- ◯ Good
- ◯ Poor

13) Did you discover any major faults in any of the above main component functionalities? *

- ◯ Yes
- ◯ No

14) If the answer you gave is "Yes" for the previous question (14), please choose the component(s) where the fault(s) was discovered.

- ☐ Recommendation System
- ☐ Price Prediction System
- ☐ Fraud Content Identification System
- ☐ Erroneous Image Identification System

15) If the answer you gave is "Yes" for the previous question (14), please state the fault(s) encountered.

Your answer

16) Please provide your overall system review for our Tievs classified advertising system. *

- ◯ Excellent
- ◯ Very Good
- ◯ Good
- ◯ Poor
- ◯ Very poor

17) Would you give your preference to recommend Tievs for others in future? *

- ◯ Yes
- ◯ No

18) If you may, you are free to provide some feedback regarding future refinements for Tievs.

Your answer

Thank you for your valuable contribution. Have a nice day!

Submit Clear form

Appendix C: Turnitin Plagiarism Report

Final Report v1.0

ORIGINALITY REPORT

7%	5%	3%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sri Lanka Institute of Information Technology Student Paper	3%
2	Najeebullah Lashari, Tarek Ganat, Daniel Otchere, Shams Kalam, Imtiaz Ali. "Navigating viscosity of GO-SiO2/HPAM composite using response surface methodology and supervised machine learning models", Journal of Petroleum Science and Engineering, 2021 Publication	1%
3	dl.lib.mrt.ac.lk Internet Source	<1%
4	link.springer.com Internet Source	<1%
5	www0.cs.ucl.ac.uk Internet Source	<1%
6	authors.library.caltech.edu Internet Source	<1%
7	Rumane, . "Front Matter", Industrial Innovation, 2013.	<1%

	Publication	
8	d.researchbib.com Internet Source	<1%
9	Nitis Monburinon, Prajak Chertchom, Thongchai Kaewkiriya, Suwat Rungpheung, Sabir Buya, Pitchayakit Boonpou. "Prediction of prices for used car by using regression models", 2018 5th International Conference on Business and Industrial Research (ICBIR), 2018 Publication	<1%
10	etds.lib.ncku.edu.tw Internet Source	<1%
11	hdl.handle.net Internet Source	<1%
12	www.diva-portal.org Internet Source	<1%
13	Submitted to UT, Dallas Student Paper	<1%
14	www.ijrte.org Internet Source	<1%
15	gigadom.in Internet Source	<1%
16	mmuratarat.github.io Internet Source	<1%

17	Ning Sun, Hongxi Bai, Yuxia Geng, Huizhu Shi. "Price evaluation model in second-hand car system based on BP neural network theory", 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017 Publication	<1 %
18	Submitted to Queen's University of Belfast Student Paper	<1 %
19	docplayer.es Internet Source	<1 %
20	repository.umac.mo Internet Source	<1 %
21	Submitted to Arkansas Tech University Student Paper	<1 %
22	Submitted to KDU College Sdn Bhd Student Paper	<1 %
23	Submitted to National College of Ireland Student Paper	<1 %
24	Submitted to University of Leeds Student Paper	<1 %
25	aakashsorathiya.medium.com Internet Source	<1 %
26	yaolong.blog	

	Internet Source	<1 %
27	Amirreza Mahpour, Tamer El-Diraby. "Incorporating Climate Change in Pavement Maintenance Policies: Application to Temperature Rise in the Isfahan County, Iran", Sustainable Cities and Society, 2021 Publication	<1 %
28	Submitted to University College London Student Paper	<1 %
29	aut.researchgateway.ac.nz Internet Source	<1 %
30	lumsa.it Internet Source	<1 %
31	machinelearningmastery.com Internet Source	<1 %
32	www.slideshare.net Internet Source	<1 %
33	"Machine Learning in Medical Imaging", Springer Science and Business Media LLC, 2015 Publication	<1 %
34	Forrest Jehlik, Alvaro Blazquez de Mingo. "A Deep Learning Approach to Modeling a Complex Multi-variate, Temporal Thermal Problem", 2019 18th IEEE International	<1 %

	Conference On Machine Learning And Applications (ICMLA), 2019 Publication	
35	Johnathan Tai, Izzat Alsmadi, Yunpeng Zhang, Fengxiang Qiao. "Machine Learning Methods for Anomaly Detection in Industrial Control Systems", 2020 IEEE International Conference on Big Data (Big Data), 2020 Publication	<1 %
36	argon.ess.washington.edu Internet Source	<1 %
37	cs.uef.fi Internet Source	<1 %
38	doctorpenguin.com Internet Source	<1 %
39	dukespace.lib.duke.edu Internet Source	<1 %
40	isaacchanghau.github.io Internet Source	<1 %
41	paduaresearch.cab.unipd.it Internet Source	<1 %
42	profile.typepad.com Internet Source	<1 %
43	pure.ulster.ac.uk Internet Source	<1 %

44	research.library.mun.ca Internet Source	<1 %
45	theiet.lk Internet Source	<1 %
46	www.groundai.com Internet Source	<1 %
47	"Machine Learning for Predictive Analysis", Springer Science and Business Media LLC, 2021 Publication	<1 %
48	dokumen.pub Internet Source	<1 %
49	eprints.bournemouth.ac.uk Internet Source	<1 %
50	eprints.lincoln.ac.uk Internet Source	<1 %
51	irjmets.com Internet Source	<1 %
52	lup.lub.lu.se Internet Source	<1 %
53	pt.scribd.com Internet Source	<1 %
54	scholar.afit.edu Internet Source	<1 %

Appendix D: Turnitin Plagiarism Report Receipt

turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Savandi Bandaranayake
Assignment title:	Research Project Final Report
Submission title:	Final Report v1.0
File name:	IT18113532_Final_Report.pdf
File size:	8.59M
Page count:	100
Word count:	23,036
Character count:	135,199
Submission date:	08-Oct-2021 10:26PM (UTC+0530)
Submission ID:	1668855452

TIEVS – DYNAMIC CAR PRICE PREDICTION SYSTEM USING ENSEMBLE LEARNING

K.S.S Bandaranayake

(IT18113532)

BSc (Hons) in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

Copyright 2021 Turnitin. All rights reserved.