Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/95/20241765

Predicting smartphone prices using machine learning algorithms

Zheng Zhao

Xiamen University Malaysia, 43900 Sepang, Selangor Darul Ehsan, Malaysia

CST2209182@xmu.edu.my

Abstract. This study investigates the prediction of smartphone prices using machine learning algorithms, motivated by the increasing demand for accurate pricing information in the highly competitive and rapidly evolving mobile phone market. With the proliferation of smartphones and the constant introduction of new models with varying specifications, it is crucial for consumers, retailers, and manufacturers to have a reliable method for estimating prices based on technical features. Accurate price prediction can assist consumers in making informed purchasing decisions, retailers in setting competitive prices, and manufacturers in optimizing production costs and pricing strategies. Utilizing Decision Tree Regression, Support Vector Regression (SVR), Random Forest Regression and Convolution Neural Network (CNN), the research investigates the efficacy of these methods in estimating prices accurately. Principal Component Analysis (PCA) is applied to reduce dimensionality and enhance model performance. Through rigorous experimentation and hyperparameter tuning, Random Forest Regression emerges as the most effective method, achieving the lowest Mean Squared Error (MSE) and the highest R-squared (R^2) score. The findings highlight the potential of ensemble learning techniques in addressing complex prediction tasks.

Keywords: Smartphone Pricing Prediction, Decision Tree Regression, Support Vector Regression, Random Forest Regression, Convolution Neural Network.

1. Introduction

In today's rapidly evolving technological landscape, smartphones have become an indispensable part of our daily lives. However, with a myriad of specifications and configurations available, accurately assessing the market value of a smartphone can be a challenging task for consumers. The constant introduction of new models and technological advancements further complicates this process. To address this challenge, this essay explores the development of predictive models that leverage machine learning algorithms to estimate the price of a smartphone based on its key features. By providing a reliable tool for understanding the pricing dynamics in the smartphone market, these models aim to empower consumers, retailers, and manufacturers.

Machine learning, as defined by [1], encompasses automatic techniques for making accurate predictions based on past observations. In this study, we utilize several machine learning algorithms, including Decision Tree Regression, Support Vector Regression (SVR), Random Forest Regression, and Convolutional Neural Network (CNN), to investigate the relationship between smartphone

DOI: 10.54254/2755-2721/95/20241765

specifications and their corresponding prices. These algorithms are chosen due to their proven effectiveness in handling regression tasks and capturing complex patterns within datasets.

To enhance the overall performance of our predictive models, we undertake a comprehensive preprocessing phase that includes not only data cleaning to remove any inconsistencies that may skew the results and some feature columns with string type of values that can not be directly converted into numeric values, but also data filling that fills in any missing values to ensure completeness and reliability of the dataset. Additionally, Principal Component Analysis (PCA) was utilized as a dimensionality reduction technique that streamlines the data by eliminating redundant features, thus improving the efficiency and accuracy of our models. Furthermore, this research endeavors to construct a novel Convolutional Neural Network (CNN) model tailored to the specific structure of our dataset. This customized CNN architecture aims to capture intricate patterns and relationships within the data, potentially leading to improved predictive performance. Through rigorous experimentation and hyperparameter tuning, we systematically evaluate the performance of each algorithm, including our novel CNN model, using Mean Squared Error (MSE) and R-squared (R^2) scores to ensure robust and accurate model assessment.

Our findings reveal that Random Forest Regression outperforms the other algorithms, achieving the lowest MSE and the highest R^2 score, while the CNN regression model ranks second in terms of performance, demonstrating their robustness and effectiveness in handling intricate prediction tasks. Overall, this study not only enhances our understanding of machine learning algorithms and their application in the smartphone market but also provides a valuable resource for consumers, retailers, and manufacturers. The developed predictive models enable data-driven decisions and help stay competitive in the ever-evolving smartphone industry.

2. Research Methods

2.1. Principal Component Analysis (PCA)

Predicting mobile phone prices based on their configurations involves managing a large number of features, each contributing to the overall value. However, incorporating all these features directly into a predictive model can lead to complexities such as overfitting, high computational costs, and decreased accuracy due to redundant or highly correlated variables. PCA addresses these issues by reducing the dimensionality of the data while preserving its essential information. Mathematically, PCA is an orthogonal linear transformation that transforms the data to a new coordinate system [2]. According to [3], PCA refers to the problem of fitting a low dimensional affine subspace S of dimension $d \ll D$ to a set of points $\{x_1, x_2, \ldots, x_N\}$ in a high-dimensional space R^D.

Specifically, PCA begins by normalizing the features to ensure equal contributions, then computes the covariance matrix to capture linear relationships. The heart of PCA lies in calculating the eigenvectors and eigenvalues of this matrix, where eigenvectors represent directions of maximum variation and eigenvalues quantify their significance. By carefully selecting the few principal components, namely the eigenvectors corresponding to the largest eigenvalues, PCA transforms the original, potentially high-dimensional feature space into a novel, condensed space that efficiently encapsulates the majority of the inherent variability within the data. This process simplifies the predictive modeling of mobile phone prices by reducing the multitude of configuration features to a few meaningful principal components, enhancing model efficiency, interpretability, and reducing the risk of overfitting.

In this research aimed at predicting the price of smartphones based on their various features and configurations, the number of principal components is considered a hyperparameter that requires tuning. By varying it, this research seeks to find the optimal number of principal components that strike a balance between retaining essential information and minimizing redundancy, thereby enhancing the performance of the predictive models. The PCA-transformed data is then converted into a pandas DataFrame for further analysis and model training, facilitating the development of robust predictive models that can accurately estimate the price of smartphones based on their configurations.

DOI: 10.54254/2755-2721/95/20241765

2.2. Decision Tree Regression

Decision Tree Regression, an adaptable extension of the classic Decision Tree algorithm traditionally utilized for classification, excels in handling regression tasks as well. Its core functionality revolves around recursively partitioning the dataset into increasingly homogeneous subsets, with the objective of minimizing the variability or error within each subset. In the tree-growing phase the algorithm starts with the whole data set at the root node. The data set is partitioned according to a splitting criterion into subsets. This procedure is repeated recursively for each subset until each subset contains only members belonging to the same class or is sufficiently small [4]. This process is guided by criteria such as minimizing the Mean Squared Error (SSE). This selection is typically achieved through an exhaustive search or a heuristic approach that aims for the most effective split. Upon reaching the stopping point, the terminal nodes, or leaf nodes, contain predictions for the target variable, which are typically the mean or median of the target values within the subset represented by each leaf. In the second phase, the tree-pruning phase, the full grown tree is cut back to prevent over-fitting and to improve the accuracy of the tree. An important approach to pruning is based on the minimum description length (MDL) principle [4].

To achieve optimal trade-off between model accuracy and complexity in decision trees, pruning algorithms systematically proceed through an iterative process of subtree or split node removal. During each iteration, the algorithm evaluates the change in Total Description Length (TDL), which encapsulates both the Data Description Length (DDL) and the Model Description Length (MDL). If the removal of a subtree or split results in a reduction of TDL, signifying an overall enhancement in the balance between fitting the data and maintaining model simplicity, then the modification is accepted. This iterative pruning process continues until no further reductions in TDL can be achieved, thereby ensuring that the resultant decision tree possesses the desired properties of both predictive accuracy and minimal complexity.

One of the most significant advantages of Decision Tree Regression lies in its interpretability. The model's tree structure offers a clear and intuitive visualization of the prediction process, revealing the importance of different features and how they interact to influence the target variable. Furthermore, the interaction level of tree-based approximations is limited by the tree size (J). Namely, no interaction effects of level greater than (J−1) are possible [5]. The depth of a feature's appearance in the tree, coupled with the error reduction achieved by splitting on that feature, serves as a quantitative measure of its significance. This insight not only enhances understanding of the data but also informs subsequent modeling strategies, such as feature selection and model simplification. In conclusion, Decision Tree Regression simplifies complex datasets by segmenting them into simpler, more understandable subsets. This partitioning process not only facilitates accurate predictions but also uncovers hidden relationships between features and the target variable, thereby providing valuable insights that can guide further data exploration and modeling efforts.

In this research, a Decision Tree Regression model is constructed utilizing the DecisionTreeRegressor class from a machine learning library, with a specified random state of 42 to ensure reproducibility of results. The model is tuned using a hyperparameter grid, where various settings for the maximum depth of the tree, the minimum number of samples required to split an internal node, and the minimum number of samples required to be at a leaf node are explored. This grid search approach allows for the identification of the optimal combination of hyperparameters, which can enhance the model's performance in accurately predicting smartphone prices based on given features such as processor speed, memory, camera quality, and other configurations.

2.3. Support Vector Regression (SVR)

SVR can be seen as a SVLA with continuous output or support vector machines for function estimation [6]. Support Vector Regression (SVR) is a sophisticated supervised learning technique tailored specifically for regression tasks, which stems from Support Vector Machine (SVM). SVMs are based on the idea of structural risk minimization (SRM) induction principle that aims at minimizing a bound on the generalization error, rather than minimizing the mean square error [7] while SVR aims to identify

DOI: 10.54254/2755-2721/95/20241765

a function that optimally maps the input features to a continuous output variable, with a particular emphasis on minimizing the prediction error while remaining resilient to the presence of outliers or noise in the dataset. Specifically, the objective of SVR is not merely to find a line or curve that passes through the data points but, instead, to construct a "tube" of width ε around the observed data. This epsilon-tolerance zone defines a region where the model is allowed to deviate from the exact values without incurring a penalty. Consequently, SVR focuses on minimizing the prediction error only for data points that lie outside this epsilon-insensitive tube, allowing it to ignore the influence of outliers and noise to a certain extent.

Therefore, the core idea behind SVR is to find a hyperplane or function in a high-dimensional space that best captures the trend of the data within the epsilon-insensitive zone. To achieve this, SVR employs a set of support vectors, which are data points that lie either on the boundaries of the epsilon-insensitive tube or beyond it. These support vectors play a crucial role in defining the optimal function, as they are the only data points that directly influence the model's training process. The optimization problem in SVR is typically solved using Lagrange multipliers and quadratic programming techniques, leading to a sparse solution where only a small subset of the training data (the support vectors) contributes to the final model. This sparsity not only makes SVR computationally efficient but also enhances its interpretability by focusing only on the most informative data points. By incorporating an epsilon-insensitive loss function and leveraging the power of support vectors, SVR provides a robust and flexible approach to regression analysis, particularly suitable for dealing with noisy datasets such as those derived from smartphone specifications, as it can effectively capture the underlying trends in the data while being insensitive to isolated outliers or irregularities, resulting in more reliable and accurate predictions.

In the quest to predict the price of smartphones based on their features or configurations, a SVR model is constructed using the SVR class from a relevant machine learning library. The model is meticulously tuned through a comprehensive grid search over multiple hyperparameters. Three distinct kernel types are considered: linear, radial basis function (rbf), and polynomial (poly). For each kernel, different values of the regularization parameter are explored to balance the trade-off between model complexity and generalization. Additionally, for the rbf and poly kernels, various settings for the gamma parameter, which controls the width of the kernel function, are investigated. This exhaustive grid search enables the identification of the optimal hyperparameter combination, tailored to enhance the SVR model's performance in accurately predicting smartphone prices based on various attributes.

2.4. Random Forest Regression

Random Forest Regression significantly bolsters the predictive prowess of individual decision trees through an ensemble learning paradigm. The idea of ensemble learning methods is to select a collection, or ensemble, of hypotheses from the hypothesis space and combine their predictions [8]. The core principle involves constructing an ensemble of multiple decision trees and amalgamating their predictions to produce a final, enhanced estimate. Specifically, random forests for regression are formed by growing trees depending on a random vector Θ such that the tree predictor $h(x, \Theta)$ takes on numerical values as opposed to class labels [9].

The academic underpinnings of this approach lie in two key mechanisms: bootstrap sampling and feature bagging. First, each tree within the Random Forest is grown independently on a distinct, randomly selected bootstrap sample of the original training dataset. This sampling strategy, known as bootstrap aggregating or bagging, introduces subtle variations in the data utilized by each tree. This randomness effectively decorrelates the trees, an essential factor for the ensemble's performance as uncorrelated trees tend to make complementary errors, thereby reducing the overall error of the ensemble. Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure [10]. Second, during the construction of each tree, a random subset of available features is considered at each split point. This feature bagging technique promotes diversity among the trees by ensuring that they focus on distinct

DOI: 10.54254/2755-2721/95/20241765

aspects of the data. By limiting the feature space explored at each split, different trees can learn distinct representations of the data, which in turn increases the diversity of their predictions.

The combination of bootstrap sampling and random feature selection fosters a heterogeneous ensemble of trees, each capturing unique patterns and relationships within the data. The final prediction of the Random Forest is computed by averaging the predictions of all individual trees. This aggregation process exploits the strengths of individual trees while minimizing biases and variances inherent in any single model. The inherent randomness in the ensemble construction and the averaging of errors serve to mitigate overfitting, resulting in higher predictive accuracy and improved generalization capabilities compared to standalone decision trees. In summary, Random Forest Regression harnesses the collective strength of multiple, diverse decision trees through ensemble learning, combined with bootstrap sampling and random feature selection. This approach enables it to effectively tackle complex datasets, identify salient features, and capture intricate non-linear relationships between variables, ultimately yielding more reliable and accurate regression estimates.

In the context of predicting the price of smartphones based on their features or configurations, a Random Forest Regression model is constructed using the RandomForestRegressor class, with a fixed random state of 42 to ensure consistent results. The model is tuned through a grid search over several hyperparameters to optimize its performance. Specifically, the number of trees in the forest, the maximum depth of the trees, the minimum number of samples required to split an internal node, and the minimum number of samples required to be at a leaf node are varied across a range of values. This hyperparameter tuning process aims to find the best combination of settings that enable the Random Forest Regression model to accurately predict smartphone prices based on a variety of features, such as processor type, memory capacity, camera resolution, and other relevant configurations. By refining these parameters, the research seeks to enhance the model's predictive power and robustness.

2.5. Convolutional Neural Network (CNN)

Deep learning refers to neural networks with many layers of hidden units. Such networks are able to learn complex representations that are useful for high-level abstraction and decision making [11]. Convolutional Neural Networks (CNNs) have transformed the landscape of deep learning, particularly in the realm of image recognition and analysis. Their unique architecture, consisting of convolutional layers, pooling layers, and fully connected layers, enables them to extract and learn complex patterns from raw image data. However, the versatility of CNNs extends far beyond image processing, and they have been increasingly applied to various regression problems, including the prediction of smartphone prices based on their configurations and features.

At the core of CNNs lies the convolutional layer, which performs a convolution operation on the input data using a set of learnable filters or kernels. These filters slide across the input data, performing element-wise multiplications and summing the results to produce a feature map. Each filter is applied to every location of the input data, which means that the same parameters are used across different positions of the input. Parameter sharing has allowed CNNs to dramatically lower the number of unique model parameters and to significantly increase network sizes without requiring a corresponding increase in training data [12]. This process allows the CNN to detect local patterns and features in the data, such as edges, corners, and textures in images, or in our case, relationships between smartphone features like processor speed and display resolution. The pooling layers, typically placed after the convolutional layers, serve to reduce the dimensionality of the feature maps by downsampling the data. This not only reduces the computational requirements of the network but also helps in achieving translational invariance, meaning that the network becomes less sensitive to the exact position of features in the input data. The fully connected layers, located at the end of the CNN, integrate the information from the preceding layers and perform the final prediction task.

In the context of this research, a novel approach is adopted where each smartphone feature is treated as an input channel in a CNN. This unconventional method of viewing features as image channels harnesses the CNN's capability to uncover complex patterns and relationships within the raw data. The designed CNN regression model refrains from using pooling layers, as there is no need to diminish the

DOI: 10.54254/2755-2721/95/20241765

spatial dimensions of the 'images' represented by the features. Instead, the model integrates batch normalization layers, which assist in mitigating the issue of internal covariate shift. This, in turn, accelerates the training process and bolsters the network's stability. Following the convolutional and batch normalization layers, fully connected layers are employed to process the output and predict the market price of a smartphone.

3. Evaluation (Programming Experiments)

3.1. Dataset Analysis

In this research, a comprehensive dataset that encompasses detailed configurations and prices of 980 diverse smartphones is utilized, which is available at https://www.kaggle.com/datasets/abdurrahman22224/smartphone-new-data. This dataset features a multitude of attributes, including brand_name, price, rating, along with various technological specifications such as 5G compatibility (has_5g), NFC and IR blaster presence (has_nfc, has ir blaster), processor brand and specifications (processor_brand, num_cores, processor_speed), etc.

It is important to note that the dataset comprises a mix of data types, including string-type features that represent categorical variables, as well as numerical features. However, not all numerical columns are free from missing values, necessitating the need for data preprocessing steps to handle these lost values. Additionally, some of the string-type features may require encoding, transformation or even deletion to be compatible with the machine learning algorithms that will be employed for price prediction. This preprocessing stage is crucial to ensure that the data is in a suitable format for model training and evaluation, ultimately leading to more accurate and reliable predictions. A segment of the dataset, featuring the label column highlighted in blue, is showcased as follows:

Table 1. Part of the dataset with feature name row and label column highlighted in gray and blue.

brand_ name	price	rating	has_5g	has_nfc	has_ir_ blaster	processor_ brand	num_ cores	processor_ speed
oneplus	54999	89	TRUE	TRUE	FALSE	snapdragon	8	3.2
oneplus	19989	81	TRUE	FALSE	FALSE	snapdragon	8	2.2
samsung	16499	75	TRUE	FALSE	FALSE	exynos	8	2.4
motorola	14999	81	TRUE	FALSE	FALSE	snapdragon	8	2.2
realme	24999	82	TRUE	FALSE	FALSE	dimensity	8	2.6
samsung	16999	80	TRUE	TRUE	FALSE	snapdragon	8	2.2
apple	65999	81	TRUE	TRUE	FALSE	bionic	6	3.22
xiaomi	29999	86	TRUE	FALSE	TRUE	dimensity	8	2.6
nothing	26749	85	TRUE	TRUE	FALSE	snapdragon	8	2.5

3.2. Library Imports

To elevate predictive modeling endeavors for smartphone pricing, a multifaceted approach that seamlessly integrates traditional machine learning with deep learning methodologies has been adopted. By leveraging a diverse set of professional-grade Python libraries, a robust modeling pipeline capable of handling the complexities of our structured dataset has been constructed .

One foundation rests upon the ubiquitous ‘numpy’ and ‘pandas’ libraries, which enable meticulous data manipulation and analysis, ensuring that the data is meticulously prepared for modeling. To refine the feature space and improve algorithm performance, ‘StandardScaler’ from ‘sklearn.preprocessing’, standardizing the features in a professional manner, is employed. Beyond conventional regression techniques, this research has embarked on a deep learning trajectory by integrating PyTorch, a state-of-the-art deep learning framework. By transforming structured smartphone features into ‘feature images’, CNN have been innovatively adapted to this unique regression problem, harnessing its prowess in identifying intricate patterns and relationships. This CNN architecture was meticulously crafted using

DOI: 10.54254/2755-2721/95/20241765

'torch.nn' for defining custom layers and 'torch.optim' for optimizing model training. Moreover, professional practices such as utilizing 'nn.BatchNorm2d' for batch normalization have been adopted, mitigating internal covariate shift and expediting convergence. Furthermore, 'DataLoader' and 'TensorDataset' from 'torch.utils.data' have been embraced for efficient data loading and batching, resulting in significantly reduced training times.

To optimize the performance of these models, hyperparameter tuning strategies have been extended to encompass both traditional machine learning models using 'GridSearchCV' from 'sklearn.model_selection' and the deep learning counterparts, where various optimizer configurations, learning rates, and training hyperparameters are experimented. In evaluating the performance of these models, a professional focus by adhering to industry-standard metrics such as 'mean_squared_error' and 'r2_score' from 'sklearn.metrics' was maintained. This approach enables the research to quantify the accuracy and goodness of fit of regression models in a consistent and comparable manner, allowing it to objectively assess the merits of both traditional and deep learning-based solutions.

3.3. Data Preprocessing

To prepare the smartphone dataset for predictive modeling, a rigorous series of data preprocessing steps were meticulously executed. These measures included first selecting and filtering out any irrelevant or non-numeric columns, as they could hinder the model's ability to correlate smartphone features with prices accurately. Utilizing a custom 'can_convert_to_numeric' function, non-numeric columns were identified and excluded, leaving only numeric data for analysis. Next, missing values within the numeric columns were identified and handled by filling them with the median value of each respective column. This strategy mitigated the impact of incompleteness on the dataset by providing a robust estimate of the "typical" value for each feature. Finally, feature scaling was performed using the 'StandardScaler' to normalize the distribution of continuous variables. By setting the mean of each feature to 0 and its standard deviation to 1, the algorithm's sensitivity to the scale of input variables was mitigated, ensuring stability and convergence during the modeling process. These preprocessing steps laid a solid foundation for building robust and accurate predictive models capable of correlating smartphone features with prices. After undergoing the aforementioned data preprocessing steps, the dataset has been refined as detailed below:

	price	rating	has_5g	has_nfc	has_ir_blaster	num_cores
0	54999	89.0	True	True	False	8.0
1	19989	81.0	True	False	False	8.0
2	16499	75.0	True	False	False	8.0
3	14999	81.0	True	False	False	8.0
4	24999	82.0	True	False	False	8.0
..	...	...	...	...	...	...
975	34990	83.0	True	False	False	8.0
976	14990	75.0	True	False	False	8.0
977	28990	85.0	True	True	True	8.0
978	19990	80.0	True	True	False	8.0
979	24990	74.0	True	False	False	8.0

Figure 1. Dataset after preprocessing.

3.4. Model Construction

In the domain of regression analysis, traditional models such as Decision Tree Regression, Support Vector Regression, and Random Forest Regression stand out for their simplicity of implementation and efficient training procedures. These algorithms, conveniently packaged in popular Python libraries like scikit-learn, facilitate regression tasks through user-friendly interfaces, allowing for straightforward model initialization, fitting to training data, and performance optimization via parameter tuning. The

DOI: 10.54254/2755-2721/95/20241765

construction process typically entails initializing the model configuration, exposing it to the training dataset, and iteratively refining the model parameters to optimize its performance.

Conversely, the construction of a CNN regression model proves to be significantly complex when confronted with intricate structured data. In this research, our model's architecture is meticulously crafted, comprising a sequence of layers, each designed with distinct functionalities and parameters. Initially, convolutional layers are employed to extract hierarchical spatial features from the input data, while batch normalization layers contribute to model stability and regularization. Subsequently, the extracted features are flattened and fed into fully connected layers, where they undergo further processing to learn non-linear relationships between inputs and outputs. This approach is particularly useful for problems where the original input features are not very individually informative [13], thereby fulfilling the regression objective.

3.5. Hyperparameter Tuning

Ideally, if we had enough data, we would set aside a validation set and use it to assess the performance of our prediction model. Since data are often scarce, this is usually not possible. To finesse the problem, K-fold crossvalidation uses part of the available data to fit the model, and a different part to test it [5]. A rigorous grid search methodology, which is an exhaustive search method that finds the optimal solution by traversing various hyperparameter combinations [14], accompanied by five-fold cross-validation was employed to identify the optimal hyperparameter configurations for traditional regression models: Decision Tree Regression, Support Vector Regression (SVR), and Random Forest Regression. The hyperparameter tuning process exhaustively explored various combinations of tree depths, sample splits, and leaf sizes for the tree-based models. For SVR, the kernel functions and their associated parameters, such as the penalty term (C) and the kernel-specific parameters (gamma for RBF and degree for polynomial), were thoroughly searched. Additionally, the influence of dimensionality reduction was investigated by varying the number of components retained through Principal Component Analysis (PCA). This multi-faceted approach enabled us to pinpoint the specific model settings that minimized the Mean Squared Error (MSE) on the validation sets, thereby optimizing the predictive power of each model.

Transitioning from the hyperparameter tuning of traditional regression models to Convolutional Neural Networks (CNNs) for our regression task, a rigorous hyperparameter tuning process was undertaken. One key concept in machine learning is that of the loss function, which measures the penalty associated with a particular decision [11]. The loss function of this model is nn.MSELoss(). This comprehensive approach aimed to identify the most effective combination of parameters, including the number of PCA components, batch size, learning rate, and optimizer, that would minimize the Mean Squared Error (MSE) and maximize the R^2 score.

The hyperparameter tuning involved iterating over a range of values for each parameter. The number of PCA components was varied from 8 to 12, allowing us to control the dimensionality of the input data and potentially reducing overfitting. The batch size was explored between 16 and 32, influencing the trade-off between training time and the stability of gradient updates. Additionally, the learning rate was adjusted from 0.001 to 0.009 in increments of 0.004, as this hyperparameter critically determines the speed and quality of the optimization process. Finally, two popular optimizers, Adam and RMSprop, were tested to evaluate their impact on model convergence and performance. One critical consideration in our CNN hyperparameter tuning was the exclusion of the Stochastic Gradient Descent (SGD) optimizer. Although SGD is a fundamental and widely used optimizer, its straightforward update rule may not be well-suited for our dataset. The small dataset size, combined with SGD's sensitivity to the learning rate and the tendency to get stuck in local minima, can lead to unstable training and potentially result in NaN (Not a Number) values when the model's weights diverge to extreme values, indicating the numerical instability during the backpropagation process.

For each unique configuration of hyperparameters, the CNN model was trained for 40 epochs, during which the model's loss was monitored and recorded. Upon completion of the training phase, the model's predictive capabilities were assessed on a held-out test set, generating MSE and R^2 scores as metrics of

DOI: 10.54254/2755-2721/95/20241765

accuracy. These results were subsequently used to evaluate the effectiveness of each hyperparameter combination. The hyperparameter tuning process highlighted the sensitivity of CNN model performance to the chosen parameter values. By systematically exploring various configurations, we were able to identify the optimal hyperparameter settings that resulted in the lowest MSE and the highest R^2 score.

3.6. Performance Comparison

With the optimal hyperparameters and PCA dimensionality configurations secured for each model, predictive evaluations were carried out on an independent test set. The quantitative assessment encompassed both Mean Squared Error (MSE) and R-squared (R^2) metrics, providing insightful measures of the models' predictive accuracies. Subsequently, the performance of Decision Tree Regression, SVR, Random Forest Regression and CNN Regression was visually juxtaposed using bar charts (Fig. 2), showcasing their respective MSE and R^2 scores.

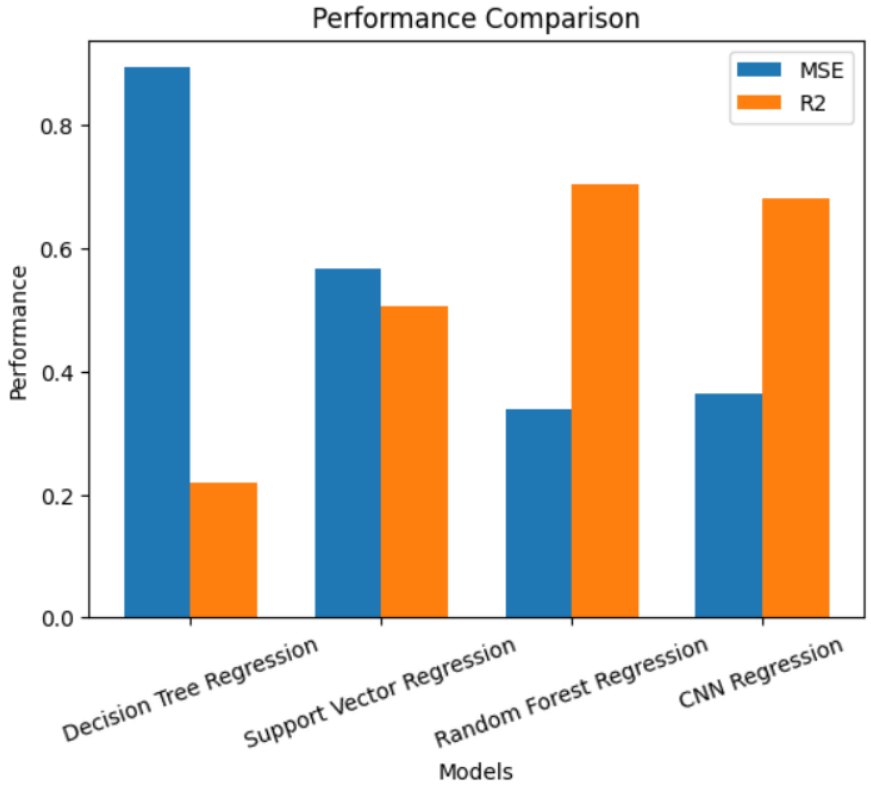

Figure 2. Performance comparison.

This comprehensive comparison underscored the predictive prowess of Random Forest Regression, as evidenced by its lowest MSE and highest R^2 values among the tested models. This outcome firmly established Random Forest Regression as the superior choice for predicting smartphone prices, demonstrating its robustness and effectiveness in capturing the intricate relationships within the data. However, it is noteworthy that the CNN model demonstrated a predictive prowess that was marginally inferior to Random Forest Regression yet substantially surpassed that of Decision Tree Regression and SVR. This result highlights the potential of CNNs for regression tasks, particularly when tailored with appropriate hyperparameters and dimensionality reduction techniques.

4. Conclusions

The comprehensive analysis and modeling efforts undertaken in this study conclusively demonstrate the predictive superiority of Random Forest Regression in accurately estimating smartphone prices.

DOI: 10.54254/2755-2721/95/20241765

Strategic feature engineering, rigorous hyperparameter tuning, and dimensionality reduction through Principal Component Analysis (PCA) have contributed significantly to enhancing model performance.

Although Decision Tree Regression demonstrated a basic level of predictive ability with an MSE of 893311633.272964 and an R^2 score of 0.21915080409112653, and Support Vector Regression (SVR) displayed moderate improvement with an MSE of 565896508.4680083 and an R^2 of 0.5053463795317432, both models were surpassed by Random Forest Regression and CNN regression model.

Random Forest Regression, with its minimized MSE of 337297421.95838773 and the peak R^2 score of 0.7051662478038455, emerged as the paramount model for this predictive task. Its optimal configuration, consisting of 100 estimators, a tree depth of 10, alongside meticulously tuned minimum leaf and split sizes, and a reduction to 12 PCA components, underscores the strength of ensemble learning in tackling intricate prediction challenges. Naturally, it is anticipated that the optimal model will exhibit the largest number of estimators and tree depth, as this will enhance the complexity of the model. However, the bias-complexity tradeoff states that choosing a more complex hypothesis class can decrease the bias but increases the risk of overfitting, while choosing a less complex class may decrease the potential for overfitting but increase the bias [15].

Notably, the Convolutional Neural Network (CNN) regression model, with an MSE of 364982944.0 and an R^2 score of 0.6809661388397217, did not outperform Random Forest Regression in this specific context, its performance is still commendable and indicative of CNNs' significant potential in regression tasks. The CNN's performance serves as a testament to the model's capability in capturing complex data patterns and relationships, suggesting its applicability in diverse regression-based domains with further refinement and adaptation. Overall, this study underscores the immense potential of sophisticated machine learning methodologies in practical applications and hints at the prospects of exploring more advanced algorithms and innovative feature engineering techniques in future research endeavors, potentially leading to even more precise and insightful predictive models.

References

[1] Schapire, R. E. (2001). The boosting approach to machine learning: an overview. *Springer New York*.

[2] Dong, G. , & Liu, H. . (2018). Feature engineering for machine learning and data analytics.

[3] Vidal, R., Ma, Y., & Sastry, S. S. (2016). Generalized Principal Component Analysis (pp. 25-34). In *Interdisciplinary Applied Mathematics (Vol 40)*. https://doi.org/10.1007/978-0-387-87811-9. Springer.

[4] Sattler, K. U., & Dunemann, O. (2001). SQL database primitives for decision tree classifiers. In *the tenth international conference*. ACM.

[5] Hastie, T., Tibshirani, R., & Friedman, J. (2017). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (pp. 295-622). In *Springer Series in Statistics*. Springer

[6] Cao, F., & Yuan, Y. (2011). Learning errors of linear programming support vector regression. In *Applied Mathematical Modelling* (pp. 1820-1828).

[7] Wang, L. (2005). Support Vector Machines: Theory and Applications. In *Machine Learning and Its Applications, Advanced Lectures. Springer-Verlag.*

[8] Russell, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (pp. 713-753). Pearson Education.

[9] Breiman, Leo. (2001). Random Forests. In *Machine Learning* (pp. 5–32). Kluwer Academic Publishers.

[10] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R (pp. 303-366). In *Springer Texts in Statistics*. https://doi.org/10.1007/978-1-4614-7138-7. Springer.

[11] Bishop, C. M. (2006). Pattern Recognition and Machine Learning (pp. 225-595). In *Information Science and Statistics.* Springer.

[12] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning (pp. 330-373). MIT Press.

DOI: 10.54254/2755-2721/95/20241765

[13] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective (pp. 995-1005). MIT Press.
[14] Chollet, F. (2017). Deep Learning with Python (pp.108-291). Manning Publications.
[15] Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms (pp. 212-217). Cambridge University Press.