COMP1816 - Machine Learning Coursework Report

Cristian Dumbravanu - 001167783
Word Count: 1962

1. Introduction

The problem was to develop optimal regression and classification prediction models using Californian housing and Titanic databases, respectively. Regressive algorithms included ElasticNet, Linear, Polynomial, and Random Forest, while classification algorithms comprised Decision Tree, KNN, Neural Networks, and SVM. Results showed Random Forest Regression outperforming others in regression, showing lower errors and higher R2 scores. In classification, Decision Tree performed robustly, followed closely by KNN and Neural Networks, with decent performance from SVM.

2. Regression

To implement the Regression models, the 'sklearn.linear_model library from scikit-learn was utilized, providing a comprehensive set of tools for fitting, predicting, and evaluating regression models, while for Random Forest Regression, the RandomForestRegressor class from the sklearn.ensemble module was employed.

Models used and their reasoning :

- Linear Regression was chosen for its simplicity, making it suitable for initial dataset exploration and it's ability to produce results when the relationship between variables is approximately linear.
- ElasticNet Regression was chosen for its capability to handle multicollinearity of the California Housing dataset by combining Lasso and Ridge regularization techniques to penalize irrelevant features while preserving the ability to capture important patterns.
- Polynomial Regression was chosen for its ability to capture nonlinear relationships between features, providing a more flexible model which can better fit the complexities in the dataset and potentially improve the predictions.
- Random Forest Regression was selected for its effectiveness in handling nonlinear relationships, particularly beneficial for the potentially complex patterns in the California Housing dataset, by having multiple decision trees trained on different data subsets to deliver robust predictions.

Metrics used and reason of application :

- Mean Absolute Error calculated the average differences between predicted and actual values, offering a straightforward measure of the average error magnitude regardless of direction,thus providing a simple and intuitive evaluation of model performance.
- Median Absolute Error measured the median of differences between predicted and actual values but less influenced by outliers compared to Mean Absolute Error and provided a more reliable evaluation of model performance.
- Mean Squared Error calculated the average of squared differences between predicted and actual values, emphasizing large errors more significantly and providing a comprehensive measure of overall model performance in terms of error spread.
- R-squared quantified the proportion of predictable variance in the target variable from the independent variables and offering insights into the model's ability to explain variability from dependent variable.

- Root Mean Squared Error, the square root of the average of the squared differences between predicted and actual values, interpretable in the same units as the target variable, and offered a clear understanding of the typical error magnitude of predictive models.

2.1. Pre-processing

The Californian Housing Dataset contains relevant records of geographical location, characteristics and the value of Californian properties.

The pre-processing of the database involves :

- 1. Using Pandas, load the Housing database and set the target(Y) to House Value and the rest of the features into a list(X).
- 2. Drop the unnecessary feature "No" as it serves as an index for the records and does not provide any meaningful information for predicting the target.
- 3. The data base consisting of 1000 records was allocated into a 800:200 ratio for training and testing using the split function of Sklearn Library.
- 3. Using the function select_dtypes(), the features were separated into numerical and categorical types.
- 4. Using the SimpleImputer from scikit-learn, missing numerical values were imputed with the mean of all train numerical features(SimpleImputer, strategy set to "mean"), while missing categorical features were imputed with the most frequent categorical record in the training split(SimpleImputer, strategy set to "most_frequent").
- 5. Imputed Categorical features were transformed into numerical format through one-hot encoding to satisfy machine learning model compatibility.
- 6. The imputed numerical and the encoded categorical features are concatenated to create a final feature matrices for training and testing.
- 7. To prevent any features from dominating the learning process, the feature values were scaled to have a mean of 0 and a SD of 1 using standardisation
- 8. Subsequently, a regression model was trained using the provided training data and applied to make predictions on the testing split.

2.2. Methodology

Random Forest regression stood out as the most effective model with its superior performance across all metrics. Notably, in training, it achieves the lowest MAE of 30,002.8599 and RMSE of 44,403.7161, indicating its ability to make predictions with the least difference from actual values. Additionally, it outperforms other models with the highest R squared score of 0.8512, demonstrating its superior ability to explain the variance in the target variable. Furthermore, Random Forest exhibits the lowest MAE of 43,801.4619 and RMSE of 60,527.9193 on the testing set, affirming its robustness and generalization capability compared to the baseline models.

Random Forest has the ability to handle complex relationships, resistance to overfitting and high predictive accuracy. Utilizing bootstrapping, Random Forest generates multiple subsets of the training data, training each decision tree on a different subset to introduce diversity and mitigate overfitting. Additionally, by considering only a random subset of features at each split in the decision tree, RF further increases model robustness and prevents overfitting, ultimately yielding more accurate predictions by averaging the outputs of all decision trees in the ensemble.

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} h_i(x)$$

Where:

- $\hat{y}$ is the predicted value.
- N is the number of trees in the forest.
- $h_i(x)$ is the prediction of the i^{th} decision tree.

2.3. Experiments

2.3.1. Experimental Settings

The baseline regression models :

- ElasticNet :
  $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$
- Polynomial :
  $\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + ... + \beta_d x^d$
- Linear :
  $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$

For hyperparameter tuning, each model underwent grid search with cross-validation to identify the best combination of parameters. For the ElasticNet model, parameters like alpha, l1_ratio, max_iter, and tol were explored. In the case of Linear Regression, the fit_intercept parameter was tuned. The Polynomial Regression pipeline was optimized for the polynomial degree. Finally, the Random Forest was fine-tuned for parameters such as n_estimators, max_depth, min_samples_split, and min_samples_leaf.The best performing hyperparameters were selected based on the metrics previously mentioned.

2.3.2. Results

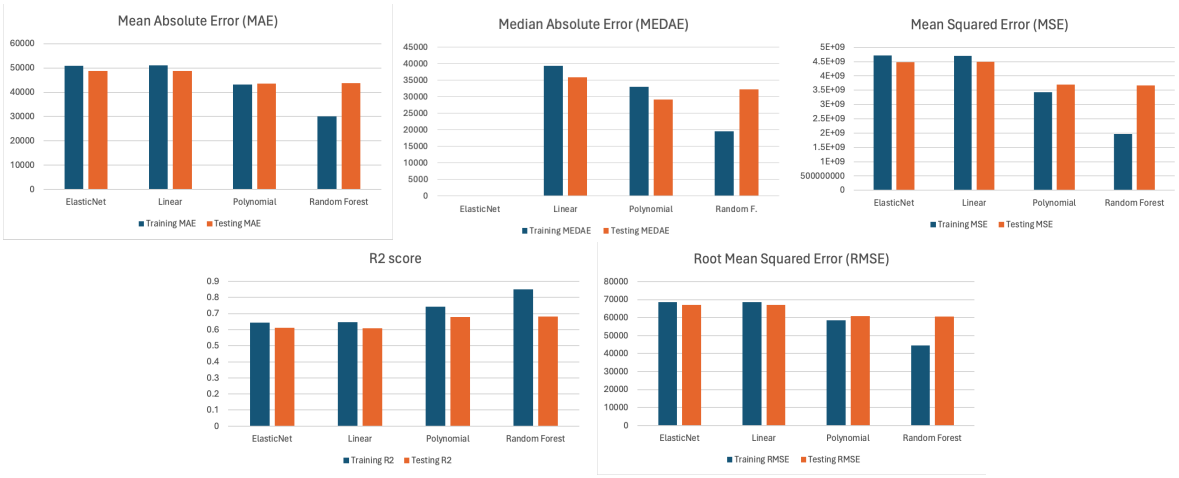

Figure 1. Results of the Regression Metrics

2.3.3. DISCUSSION

Random Forest excelled in regression, displaying the lowest MAE of 30,002.8599 on training and 43,801.4619 on testing data, along with the highest R-squared scores of 0.8512 and 0.6817, respectively, indicating superior accuracy and fit. It also showed the lowest median absolute error on both training (19,591.9340) and testing (32,166.8570) data, demonstrating robustness against outliers and better generalization. While Elastic Net showed moderate performance, with slightly better MAE but marginally worse RMSE and R2 scores compared to Linear Regression, Polynomial Regression outperformed all baselines with superior R2 score, lower MAE and RMSE, suggesting higher accuracy and variability explanation.

3. Classification

For classification, Decision Tree, KNN, Neural Network, and SVM were implemented using scikit-learn's respective modules.

Models used and their reasoning :

- Decision Tree Classifier was selected because of its ability to handle both numerical and categorical data and flexibility in capturing complex decision boundaries.
- K-Nearest Neighbours was applied for its simplicity and effectiveness in classifying data points based on the feature space, making it effective for non linear classification tasks.
- Neural Networks were selected for their ability to capture complex nonlinear relationships though hierarchical feature representation.
- Support Vector Machine was utilised for its ability to maximise the margin between different classes.

Metrics and reason of application :

- Accuracy : provided a simple measure of the model's performance (ratio of correctly predicted instances to the total number of instances).
- Precision : measured the ratio of correctly predicted positive observation to the total predicted positives.
- Recall : measured the ratio of correctly predicted positive observations to all observations.
- F1 score : provided a balanced measure where both precision and recall are important.
- The ROC curve : provided insight into the classifier's performance across different thresholds.
- Area Under Curve: quantified the classifier's overall performance with a single scalar value.

3.1. Pre-processing

The Titanic dataset is a modified version of the benchmark dataset containing passenger information from the infamous Titanic shipwreck, with variables including survival status, ticket class, name, age, gender, number of siblings/spouses and parents/children aboard, ticket number, fare, and port of embarkation.

The pre-processing of the database :

- 1. The CSV Titanic Dataset is read using Pandas and the relevant features are set as the independent(X) variables and the target variables survival as (Y).
- 2. The "PassengerId" and "Name" columns are both dropped as they do not offer any information that could help predict if a passenger has survived or not.
- 3. The database consisting of 890 data-points is split into a 650:240 ratio for training and testing.
- 4. The rest of the pre-processing (data type separation, imputation, standardisation, one hot encoding and continuation) is exactly the same as the one mentioned in 2.1.

3.2. Methodology

Decision Tree classification emerged as the primary model due to its notable performance across key metrics. In training, it achieved high accuracy of 89.69%, with precision and recall scores of 0.91 and 0.88, respectively, showcasing its effectiveness in correctly classifying both positive and negative instances. Similarly, on the testing set, it maintained a respectable accuracy of 84.17%, with precision and recall scores of 0.83 and 0.82, respectively, demonstrating its capability to generalize well to unseen data. Moreover, the Decision Tree model's F1-Score on the testing set was 0.82, further affirming its balanced performance in terms of precision and recall.

The Decision Tree Classification model recursively splits the data into subsets based on the feature that best separates the classes. It makes decisions by following a tree-like structure, where each internal node represents a feature, each branch represents a decision rule and each leaf node represents a class label.

$$D_{\text{left}} = \{(x, y) \in D \mid x_i \leq t\} \tag{1}$$

$$D_{\text{right}} = \{(x, y) \in D \mid x_i > t\} \tag{2}$$

3.3. Experiments

3.3.1. Experimental Settings

The baseline Classification models and equations :

- K-Nearest Neighbours :

  (x) = mode(y_i) where i $\in$ neighbors(x)

- Neural Network :

  (x) = softmax(W$\hat{}$(2)ReLU(W$\hat{}$(1)x + b$\hat{}$(1)) + b$\hat{}$(2))

- Support Vector Machine :

  Linear SVM :

  f(x) = sign(w$\hat{}$Tx + b)

  Non-linear SVM :

  f(x) = sign
  left($\sum\_i = 1\hat{N}$
  $alpha\_iy\_iK(x, x\_i) + b$
  $right)$

For hyperparameter tuning a parameter grid containing various combinations of hyper parameters is defined and using GridSearchCV, search through this parameter grid and find the best combination of hyper parameters using cross-validation. At the end using a Cross validation evaluate the model's performance on different subsets of the training data. The model, trained with the best hyperparameters from grid search is then evaluated on both training and testing datasets utilizing classification metrics mentioned earlier.

3.3.2. Results

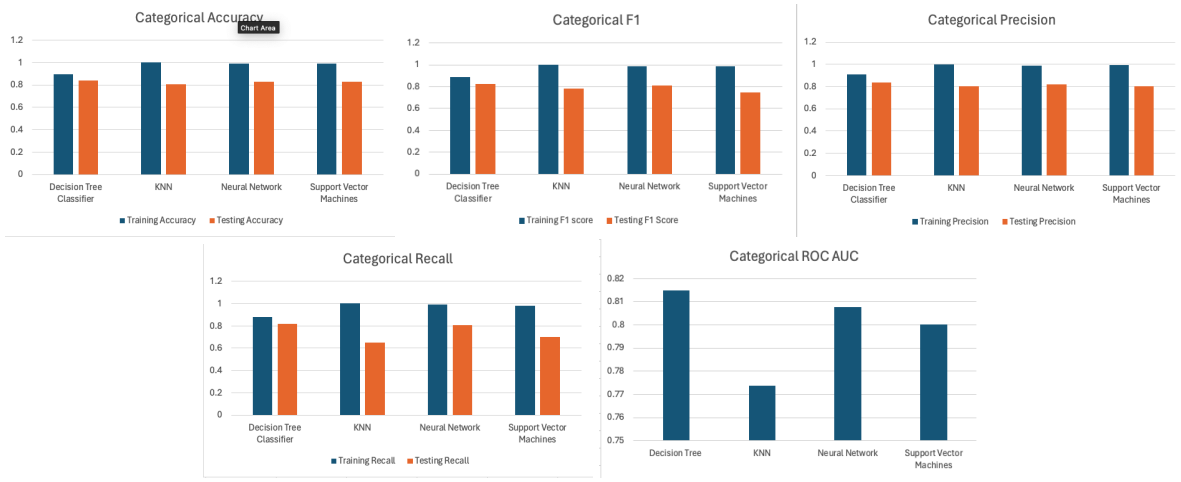

Figure 2. Results of the Categorical Metrics

3.3.3. Discussion

The Decision Tree is the main model for several reasons. Firstly, it demonstrated consistently high performance across both training and testing datasets, with accuracy rates of approximately 89.69% and 84.17%, respectively. Additionally, its precision, recall, and F1-score metrics for both classes were notably robust, indicating a balanced performance in correctly identifying both positive and negative instances. Finally, the Decision Tree model achieves a relatively high area under the ROC curve (ROC AUC) of approximately 81.50%.

In comparing the models, KNN achieved perfect accuracy of 100% on the training set but performed worse on the testing set with an accuracy of 80.83%. The Neural Network model achieved an accuracy of 98.92% on the training set and 82.92% on the testing set, demonstrating slightly better generalization capability than KNN. SVM also performed well with an accuracy of 98.92% on the training set and 82.92% on the testing set, similar to Neural Network. However, KNN showed higher precision, recall, and F1-score for both classes on the testing set, indicating its effectiveness in classification tasks despite its lower overall accuracy on the testing set.

4. Conclusion

The database analysis provided valuable insights into regression and classification prediction modeling. For regression, Random Forest outperformed ElasticNet, Linear, and Polynomial regression on the metrics MAE, RMSE, and R-squared score. In classification tasks using the Titanic dataset, Decision Tree Classifier showed impressive accuracy and balanced precision and recall. However, limitations included small dataset sizes and lack of diverse evaluation metrics. Future work could focus on increasing dataset sizes, incorporating additional metrics like Max Error and MCC, and ensuring model generalization through validation splits.