Machine Learning Methods for Urban Resource Optimization

Shu Cheng (Bernie) Chen

July 4, 2024

Abstract

This report investigates the application of predictive modeling techniques to solve regression and classification tasks using two distinct datasets. The first part focuses on predicting bike-sharing demand, utilizing the bike-sharing dataset from the UC Irvine Machine Learning Repository. Various regression models, including Ordinary Least Squares (OLS), Ridge, LASSO, Decision Tree, Random Forest, and Gradient Boosting, were employed, with Gradient Boosting showing superior performance due to its ability to capture non-linear relationships. The second part addresses office occupancy detection using Logistic Regression, Random Forest, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors (KNN), and Multilayer Perceptrons (MLP). Random Forest emerged as the most practical model due to its balance of accuracy and interpretability. Future considerations include examining the impact of preprocessing techniques, such as handling class imbalance, and exploring hyperparameter tuning on more complex datasets to enhance model performance and generalizability.

Keywords: Regression, OLS, Ridge, LASSO, Decision Trees, Random Forest, Gradient Boosting, Classification, Logistic regression, LDA, QDA, KNN, Neural Networks, MLP

1. Introduction

In data science and machine learning, applying predictive modeling techniques to real-world problems can yield highly interesting results, yet it can be complex to achieve accurately. This report explores regression and classification tasks on different datasets to solve specific issues. The first part of the report addresses the regression task, aiming to predict bike-sharing demand. Bike-sharing systems are vital for urban transportation, offering a green and convenient alternative. Predicting bike-sharing demand could help optimize resources, cut costs, and improve user satisfaction for numerous sharing companies that already exist. The second part of the report shifts focus to a classification task, specifically the classification of office occupancy detection. This report will use several classification techniques, including LDA, QDA, Random Forest, KNN, Logistic Regression, and MLP. Each method has unique strengths for handling complex data; thus, the objective is to evaluate how well different models detect occupancy, providing insights into their performance and suitability for similar tasks.

2 Bike Sharing Dataset

The bike-sharing dataset, sourced from the UC Irvine Machine Learning Repository, consists of 17,389 instances and 13 features. It records the hourly and daily count of rental bikes from the Capital Bikeshare system between 2011 and 2012, along with corresponding weather and seasonal information (Table 1). Examining different relational plots allows for better data preprocessing for more accurate regressions.

Variable	Type	Description
season	Feature	Categorical - 1: Winter, 2: Spring, 3: Summer, 4: Fall
yr	Feature	Categorical - Year (0: 2011, 1: 2012)
mnth	Feature	Categorical - Month (1 to 12)
hr	Feature	Categorical - Hour (0 to 23)
weekday	Feature	Categorical - Day of the week
weathersit	Feature	Categorical - 1: Clear, 2: Mist, 3: Light Rain, 4: Heavy Rain
workingday	Feature	Binary - If day is neither weekend nor holiday is 1, otherwise is 0
holiday	Feature	Binary - Weather day is holiday or not
temp	Feature	Continuous - Normalized temperature in Celsius.
atemp	Feature	Continuous - Normalized feeling temperature in Celsius.
hum	Feature	Continuous - Normalized humidity. Divided to 100 (max)
windspeed	Feature	Continuous - Normalized wind speed. Divided to 67 (max)
cnt	Target	Integer - Count of total rental bikes

Table 1: Description of Dataset Variables
Source: https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset

Specifically, analyzing the distributions of individual feature histograms in Figure 1. Most of the numerical features, such as `temp`, `atemp`, `hum`, and `windspeed`, exhibit normalized distributions, which is beneficial for many regression models as it ensures that no single feature disproportionately influences the model's performance. However, there are noticeable imbalances in the `weathersit` and holiday categorical variables. The target variable `cnt` shows a skewed distribution with a long tail, indicating the presence of extreme values. To address this skewness, a log transformation can be applied to the `cnt` variable. Log transformation compresses the range of the data, reducing the impact of extreme values and making the distribution more symmetric. This transformation helps in stabilizing the variance and improving the performance of regression models by making the relationship between the features and the target variable more linear.

Examining the correlation matrix (Figure 2), the employment of several feature selection and engineering methods can enhance the application of regression models to this dataset. First, due to the high correlations between `temp/atemp` and `season/mnth`, removing `mnth` will reduce multicollinearity between features.

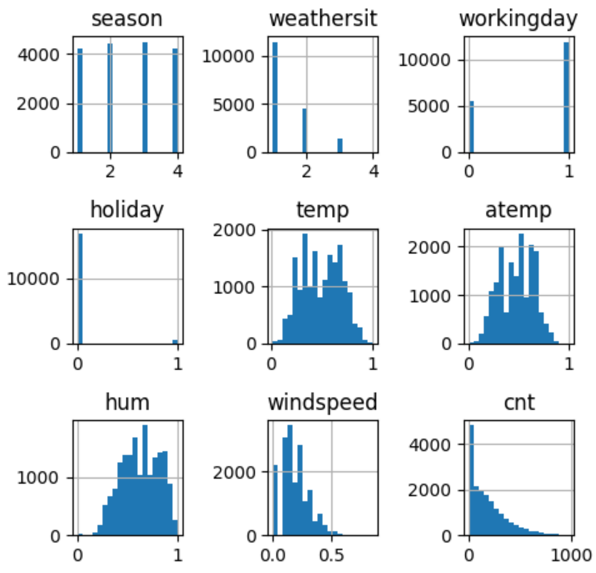

Figure 1: Distribution of Features

To better capture temporal effects, the `hr` variable will be grouped into broader time-of-day categories (night: 0-5, morning: 6-11, afternoon: 12-17, evening: 18-23). Secondly, creating interaction terms between `temp/hum`, `hum/windspeed`, `windspeed/temp`, and `atemp/temp` will reflect combined effects on the target variable. For instance, while temp and hum individually influence the number of bike rentals (cnt), their interaction (e.g., hot and

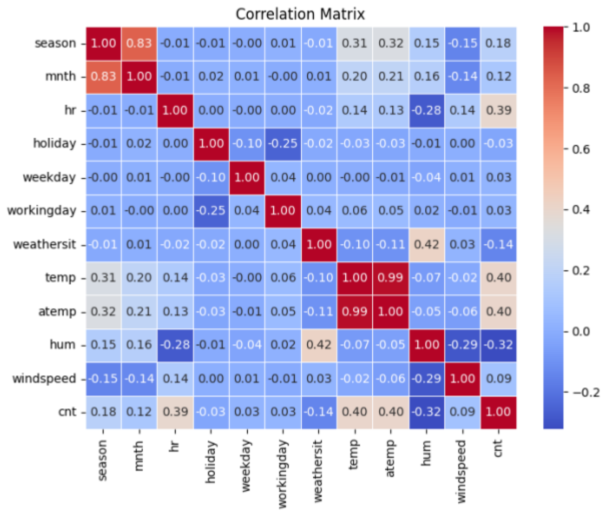

Figure 2: Correlation Matrix of Bike Features

humid conditions) might have a different and more significant effect. By introducing an interaction term such as `temp * hum`, the model can account for this combined impact. One-hot encoding categorical variables like `season`, `weathersit`, `holiday`, `weekday`, `workingday`, and the newly reclassified variable `hr` will help with better numerical representation during regression.

Lastly, looking at the pairplot in Figure 3, there is a clear seasonal pattern in `temp` and `atemp`, so including either `mnth` or `season` as a predictor variable in the regression model would help capture these non-linear seasonal effects. This further reinforces the need to remove `mnth` to prevent multicollinearity and increased model complexity. Many of the other relationships already discussed and found in the pairplot indicate significant interactions that can enhance the model's predictive power. For instance, the strong positive correlation between temp and `cnt` suggests that higher temperatures generally lead to increased counts, likely due to better weather conditions for outdoor activities. Similarly, the negative correlation between `hum` and `cnt` indicates that higher humidity might deter people from engaging in outdoor activities, thereby reducing the count. Including these variables in the model while carefully addressing potential multicollinearity can lead to a more accurate and interpretable regression analysis.

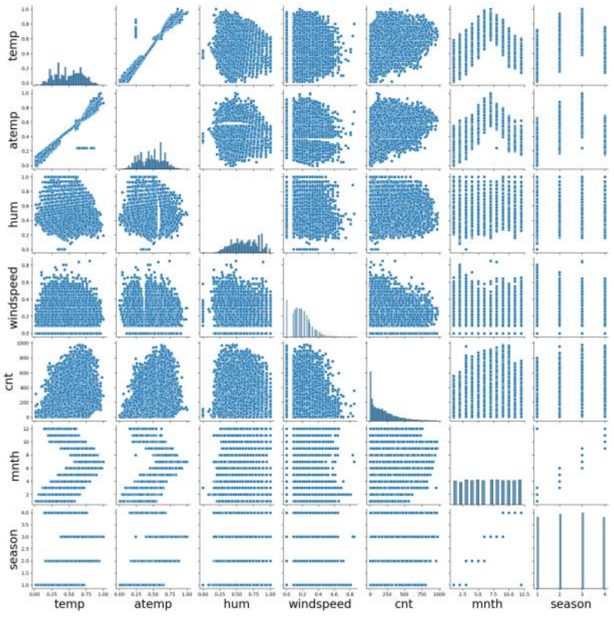

Figure 3: Pairplot of Features

3 Regression Analysis

A linear model assumes a linear relationship between the input variables (X) and the output variable (Y). This relationship can be described using a linear equation, combining input variables with coefficients (betas) that represent their contribution to the output. The linear regression model is written as:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_p X_{pi} + \epsilon_i, \quad i = 1, \ldots, n.$$

3.1 OLS Regression

Ordinary Least Squares (OLS) is the most common method for estimating these coefficients, minimizing the sum of squared differences between observed and predicted values. In this analysis, the OLS regression was performed using various predictors, including interactions between variables. Table 2 reveals the key results from the OLS regression.

Inputs	Coefficients	Std Error	P>|t|
const	4.15	0.10	-
holiday	**0.90**	0.05	-
workingday	**1.05**	0.03	-
temp	0.58	0.35	0.095
atemp	**1.19**	0.37	0.001
hum	**-1.00**	0.16	-
windspeed	-0.06	0.30	*0.846*
season_spring	-0.26	0.03	-
season_summer	-0.42	0.03	-
season_winter	-0.48	0.03	-
hr_evening	-0.18	0.03	-
hr_morning	-0.23	0.03	-
hr_night	**-2.38**	0.03	-
weekday_mon	1.06	0.03	-
weekday_sat	0.09	0.03	0.003
weekday_sun	1.14	0.03	-
weekday_thu	-0.02	0.03	0.502
weekday_tue	-0.05	0.03	0.148
weekday_wed	-0.08	0.03	0.012
weathersit_heavy_rain	-0.44	0.79	*0.581*
weathersit_light_rain	-0.38	0.04	-
weathersit_mist	0.06	0.02	0.006
hum_windspeed	-0.73	0.36	0.040
temp_hum	0.55	0.26	0.034
windspeed_temp	0.77	0.40	0.055

Table 2: OLS Regression

The OLS results reveal several significant predictors impacting the demand for bike-sharing rentals. Positive influences include holidays (0.90), working days (1.05), and apparent temperature (1.19, p=0.001). Conversely, humidity (-1.00) and certain times of day, particularly night hours (-2.38), have significant negative effects. Weather conditions also play a role, with light rain decreasing the dependent variable (-0.38), while mist has a slight positive impact (0.06, p=0.006). Interaction terms such as humidity and windspeed (-0.73, p=0.040) and temperature and humidity (0.55, p=0.034) further affect the model. To slightly improve model fit, we could consider removing non-significant indicators like windspeed (p=0.846) and heavy rain (p=0.581), focusing on variables with stronger predictive power (Table 2).

3.2 Ridge Regularization

Ridge regression is a variation of linear regression designed to address the potential issue of overfitting and reduce the model's complexity. This is achieved by modifying the loss function to include a penalty term proportional to the sum of the squares of the coefficients. The objective function for Ridge regression is given by:

$$\sum_{i=1}^{n}(y_i - \beta_0 + \sum_{j=1}^{M}\beta_j X_i^j)^2 + \lambda \sum_{i=1}^{M}\beta_i^2.$$

The penalty parameter λ controls the amount of shrinkage applied to the coefficients. As λ increases, the coefficients are pushed closer to zero, reducing the model complexity and multicollinearity among predictors. By applying this regularization to the features, this model can be compared with the other regressions (Table 3).

3.3 LASSO Regularization

LASSO regression is another form of linear regression with an objective function and penalty given by:

$$\sum_{i=1}^{n}(y_i - \beta_0 + \sum_{j=1}^{M}\beta_j x_i^j)^2 + \lambda\sum_{i=1}^{M}|\beta_i|$$

LASSO performs soft thresholding, where each coefficient is translated by λ and truncated at zero. This means that coefficients smaller than a certain threshold are set to zero, effectively removing the corresponding predictors and reducing model complexity. By setting some coefficients to zero, the model is more generalized and easier to interpret. In the context of this dataset, applying LASSO with a grid-searched optimal alpha, the model is simplified by eliminating insignificant variables such as `windspeed`, `atemp_hum`, and `weathersit_heavy_rain`, and reducing the influence of several others, especially the weekday dummies. As `weathersit_heavy_rain` has shown to be insignificant in all three linear regression models, this can be explained by the enormous imbalance observed in the `weathersit` feature. The feature imbalance can be addressed by applying sampling techniques to ensure a more balanced distribution of data for the regression model training.

	OLS	Ridge	LASSO
Lamba (λ)	-	.1	.0008697
MSE	.59421	.59421	.59429
R^2	.70517	.70517	.70513

Table 3: Linear Regression Comparison

When examining QQ plots across the three models, the residuals of the models follow a normal distribution reasonably well, suggesting that the assumptions of linear regression hold and the models are performing adequately. Overall, incorporating interaction terms, one-hot encoding, and log transformation has resulted in similar performance scores across the OLS, Ridge, and LASSO models (Table 3). These similar scores indicate that regularization does not significantly alter performance in this case, given the data and the applied transformations.

3.4 Decision Tree Regression

Decision Tree Regression works by splitting the data into subsets based on feature values, creating a tree structure where each node represents a decision rule and each leaf node represents a predicted value. The tree is built by recursively partitioning the data to minimize the mean squared error (MSE) within each subset. This method is simple to understand and visualize and can capture non-linear relationships. However, it is prone to overfitting, especially with deep trees, and exhibits high variance with small changes in data. To optimize Decision Tree Regression and mitigate overfitting, cost-complexity pruning can be employed. This involves using a complexity parameter, `ccp_alpha`, to penalize the number of nodes in the tree. Tree-based models are also less affected by the ordinality of non-dummy categorical variables, so dummy variables for categorical features were not included in the training of these models. However, a log transformation was still applied due to the skewed distribution of the target variable, `cnt`.

3.5 Random Forest Regression

Random Forest Regression, on the other hand, is an ensemble method that builds multiple decision trees and combines their predictions. Each tree is trained on a bootstrap sample of the data, and a random subset of features is considered for splitting at each node. The final prediction is the average of all tree predictions. This method reduces overfitting by averaging multiple trees, handles high-dimensional data well, and is robust to outliers and noise. In addition, hyperparameter tuning was conducted by adjusting parameters like `max_depth` to control the depth of each tree and `n_estimators` to set the number of trees in the forest. However, it is less interpretable than a single decision tree. We can still deduce some explainability by extracting feature importance, which is calculated by measuring the total reduction in MSE brought by each feature across all trees in the forest (Figure 4).

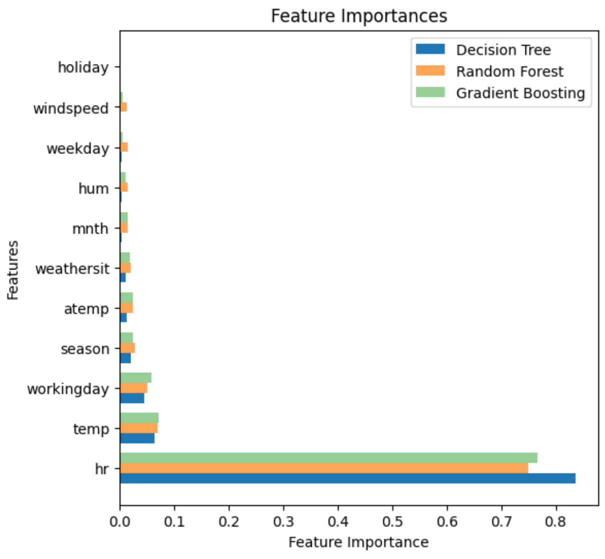

Figure 4: Tree Models Feature Importance

3.6 Gradient Boosting Regression

Gradient Boosting Regression builds an ensemble of decision trees sequentially, with each tree correcting the errors of the previous ones by fitting to the residuals of the preceding model. Tuning Gradient Boosting involves adjusting the `learning_rate`, which controls the contribution of each tree to the final model. The scatter plot indicates that the Gradient Boosting model, represented by green points, has the least scatter around the true values, suggesting the highest accuracy among the three methods. The green points are closer to the perfect prediction line (red), demonstrating the model's ability to capture the underlying data structure more effectively (Figure 5) compared to the other models, which have greater scatter. From the feature importance chart, it is clear that the hour of the day (`hr`) is the most significant feature for all three models (Figure 4).

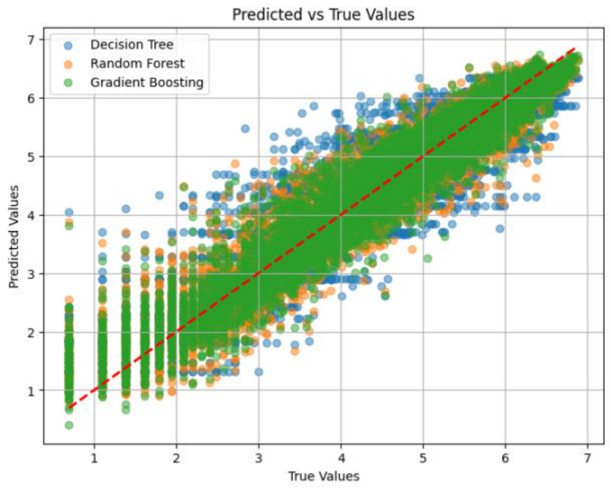

Figure 5: Tree Model Residuals

This supports findings from the linear regressions. In summary, given the tuned hyperparameters in this instance (Table 4), the Gradient Boosting model outperforms both the Decision Tree and Random Forest models in terms of prediction accuracy, as evidenced by the tight clustering of its predictions around the true values. Additionally, all three models outperform the linear regressions and regularization techniques, as indicated by their lower mean squared error (MSE) and higher R² values (Table 4). These metrics demonstrate that the tree-based models, particularly the Gradient Boosting model, provide significant improvements in prediction accuracy and robustness compared to linear regression methods when applied to this dataset.

	DT	RF	GB
MSE	. 26130	.15736	**.14142**
R^2	.87017	.92181	**.92973**
ccp/learning_rate	.00121	-	.07235
max_depth	15	15	5
n_estimators	-	108	250

Table 4: Tree-Based Regression Comparison

4 Office Occupancy Dataset

The Occupancy Detection dataset, sourced from the UC Irvine Machine Learning Repository, comprises 20,560 instances and six primary features (`Temperature`, `Humidity`, `Light`, `CO2`, `HumidityRatio`) used for predicting room occupancy (`Occupancy`). Initial preprocessing involves converting relevant columns to numeric types, handling missing values, and extracting time-related features from the timestamp, such as month, day, and hour, for additional dimensionality. By examining the distribution plot of the target variable (`Occupancy`), we observe a considerable class imbalance, with non-occupied instances far outnumbering occupied ones. This imbalance is addressed by applying `imblearn`'s `RandomOverSampler` oversampling technique to the minority class.

The correlation matrix highlights strong positive correlations between `Light` and `Occupancy`, while `HumidityRatio` and `Humidity` show moderate correlations (Figure 5).

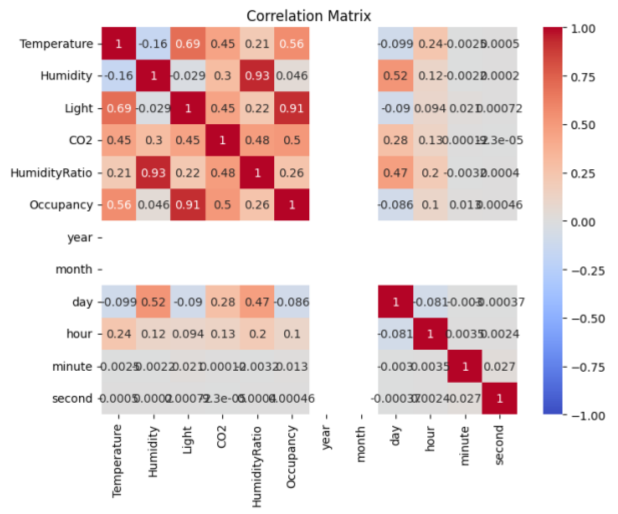

Figure 5: Correlation Matrix of Data Features

These will be treated for quadratic discriminant analysis (QDA) to prevent collinearity from affecting model performance. Despite weak correlations of temporal features with occupancy, they are retained to potentially capture periodic patterns. Normalization using a `StandardScaler` is performed to standardize the different units of the features, ensuring that all features contribute equally to the models.

5.1 Logistic Regression

Logistic regression is a binary classification technique that models the relationship between a dependent binary variable and independent variables using a logistic function, predicting the probability of an outcome. As a discriminative model, it focuses on class boundaries rather than individual class distributions. The model estimates coefficients predicting the log odds of the outcome, represented by the following equations:

$$\begin{aligned} y_i &= \text{Bernoulli}(\pi(c_k|X_i)) \\ \pi(c_k|X_i) &= \sigma(X_i\beta) \text{ or else } \log\left(\frac{\pi(c_k|X_i)}{1-\pi(c_k|X_i)}\right) = X_i\beta \end{aligned}$$

Unlike linear regression, logistic regression maximizes the likelihood of observing the sample data. In the context of the dataset, logistic regression identifies `Light` as the most significant predictor, followed by `HumidityRatio` and `Temperature`, while temporal features of `month`, `second`, and `year` are insignificant (Figure 6). The sensitivity and scoring metrics in Table 5 indicate that the logistic regression model performs exceptionally well on the given dataset. Due to the already high accuracy, further tuning was not conducted. However, regularization can be employed to prevent overfitting and improve model generalization.

5.2 Random Forest Classifier

Random forest classifiers differ from regressors by predicting categorical outcomes rather than continuous values, excelling in classification tasks. The random forest classifier on this dataset achieved superior performance metrics with extremely high accuracy. Compared to logistic regression, the random forest classifier demonstrates higher precision, F1 score, and AUC (Table 5). Both models identify `Light` as the most significant feature, but the random forest gives more importance to `Temperature` while still eliminating similar temporal features.

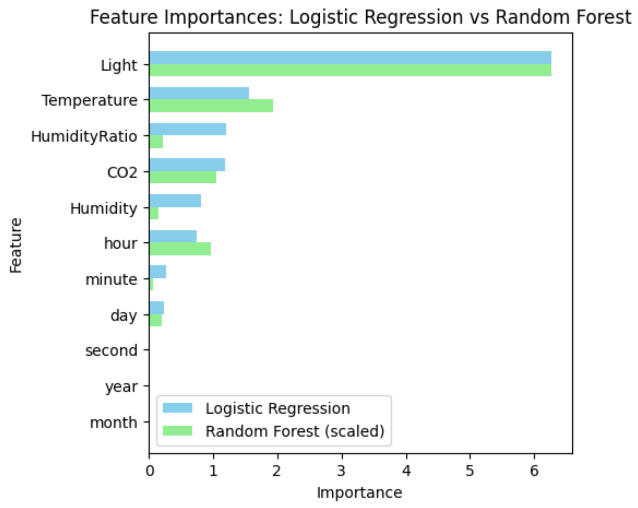

Figure 6: LR vs RF Feature Importance

The Random Forest Classifier above was trained without a depth limit, resulting in a depth of 18, which is quite complex. Despite this increased complexity, it doesn't cause the usual overfitting. This is likely because the Random Forest algorithm benefits from the ensemble approach, where multiple trees collectively make the final prediction, reducing the risk of overfitting. Tuning the `max_depth` parameter, as seen in Figure 7, shows that the accuracy is nearly the same even at a `max_depth` of 2.

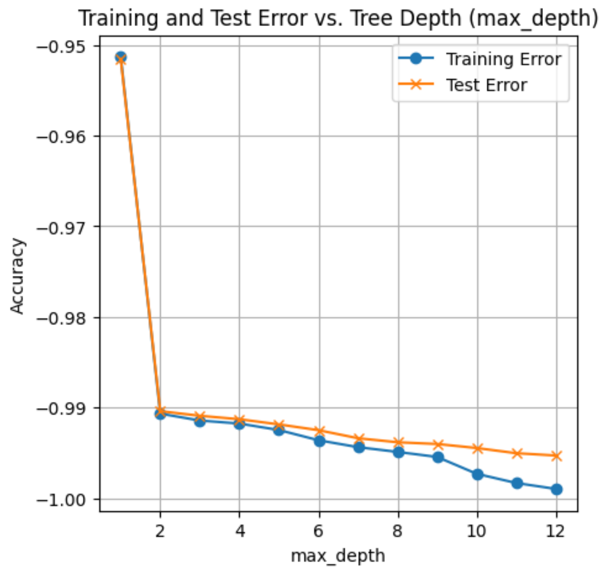

Figure 7: Random Forest Tuning

Given the minimal trade-off between depth and accuracy, selecting a shallower model could be more practical due to its enhanced interpretability (Figure 8).

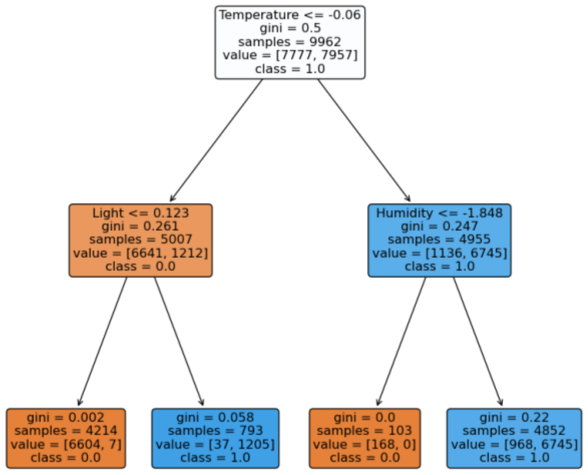

Figure 8: RF Classifier (max_depth=2)

5.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a generative model used for classification, assuming each class generates data based on a Gaussian distribution with distinct means but a shared covariance matrix. It models the distribution of predictors separately for each class. Parameter estimation can be performed using Maximum Likelihood Estimation (MLE) or Bayesian Inference. In practice, `Scikit-learn` implements LDA by fitting a linear decision boundary based on these assumptions and solving for the coefficients that maximize class separation. It demonstrates strong performance with high recall and AUC, indicating its robust ability to accurately classify the positive class within this dataset (Table 5).

5.4 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA) is similar to LDA but differs by allowing for different covariance matrices for each class. This flexibility results in a quadratic decision boundary rather than a linear one, which can help accommodate more complex relationships between the variables. When applied to the dataset, QDA initially performed less effectively due to the high collinearity among features. For instance, `HumidityRatio` correlated with `Humidity`. Removing these highly correlated features improved the model's performance. Additionally, converting temporal features such as `hour`, `day_of_week`, and `month` into dummy variables helped by allowing the model to capture non-linear relationships and interactions. Despite these adjustments, QDA did not achieve the same level of accuracy as LDA and the other models on this dataset, demonstrating the importance of feature selection and transformation in improving model performance. While QDA provides greater flexibility by allowing different covariance matrices for each class, this complexity can lead to overfitting, especially in the presence of multicollinearity.

Metric	RF	LDA	QDA	KNN	MLP	LR
Accu.	.989	.985	.895	**.991**	**.991**	.988
Prec.	.959	.941	.685	**.966**	.964	.953
Recall	.993	**.998**	1.00	.996	.997	.997
F1	.976	.969	.813	**.980**	**.980**	.975
AUC	.993	.994	.932	.998	**.999**	.994

Table 5: Model Score Comparisons

5.5 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) classifier is a non-parametric, instance-based learning algorithm that classifies data points based on the majority class among its k-nearest neighbors in the feature space. Unlike the other models, KNN makes predictions directly from the training data without constructing a generalized model. It relies on distance metrics, such as Euclidean distance, to find the closest data points and determine the class of a new instance. When applied to the dataset, KNN achieved impressive performance metrics (Table 5), performing best at k=2. This binary classification dataset benefits from KNN's simplicity and effectiveness, as the proximity-based classification works well in distinguishing the two classes based on feature similarities. However, while KNN is effective, it lacks high explainability compared to models like logistic regression or decision trees, as it does not provide insights into the underlying relationships between features and the target variable.

5.6 Multilayer Perceptron

The Multi-Layer Perceptron (MLP) classifier is a type of neural network that consists of an input layer, one hidden layer, and an output layer by default. In this dataset, the MLPClassifier applies weights and the default ReLU (Rectified Linear Unit) activation function to pass inputs through these layers, capturing non-linearity and enabling the model to learn and represent complex relationships. The ReLU activation function outputs the input directly if it is positive; otherwise, it outputs zero. This introduces non-linearity into the model, allowing it to approximate any continuous function given enough neurons and layers.

The MLP achieved the highest scores across most metrics for this dataset (Table 5). However, its complexity means it lacks the high explainability of models like logistic regression, making it harder to interpret the relationships between features and the target variable. Given that this model already performs exceptionally in prediction, there was minimal tuning involved. However, other important hyperparameters include the solver (e.g., `'adam'`, `'lbfgs'`, or `'sgd'`), which affects the optimization algorithm, and the maximum number of iterations, which controls training duration. The learning rate can also be adjusted to control the step size during gradient descent. Tuning these hyperparameters can significantly impact the model's performance, allowing it to converge faster, avoid overfitting, and better capture underlying data patterns.

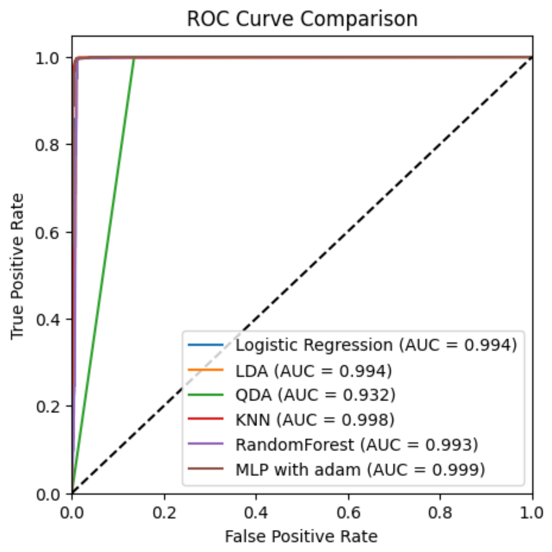

Figure 9: ROC Cure Comparison

In summary, nearly all the classification methods performed well on the Occupancy Detection dataset (Figure 9), demonstrating high accuracy and effective predictions. The Random Forest classifier stands out as the most practical model due to its excellent prediction accuracy and the clear explainability of its decision-making process. This balance of performance and interpretability makes it a strong candidate for real-world applications. While more complex models like the MLP and KNN showed slightly higher metrics, their lack of transparency compared to Random Forest makes them less practical for implementation.

Conclusions

In this report, we explored predictive modeling techniques for regression and classification tasks using the bike-sharing and office occupancy datasets. Our analysis demonstrated that incorporating interaction terms, feature selection, and transformation techniques can significantly improve regression model performance. Tree-based models, particularly Gradient Boosting, outperformed linear regression methods, showcasing their strength in capturing non-linear relationships. For classification, Random Forest and Multilayer Perceptrons excelled, with Random Forest offering a balance of high accuracy and interpretability.

Future research should delve into the effectiveness of various preprocessing techniques, such as addressing class imbalance and feature scaling, to further enhance model performance. Additionally, examining more complex datasets could provide deeper insights into hyperparameter tuning and model optimization. By exploring these avenues, we can refine predictive models to handle diverse and intricate real-world problems more effectively.

References

[1] Fanaee-T,Hadi. (2013). Bike Sharing. UCI Machine Learning Repository. https://doi.org/10.24432/C5W894.

[2] Candanedo,Luis. (2016). Occupancy Detection . UCI Machine Learning Repository. https://doi.org/10.24432/C5X01N.

[3] Kalogeropoulos, K. (2024). Lecture 3-5, 9-19: Linear Regression, Classification, Tree-based Methods [PowerPoint slides]. London School of Economics https://www.lse.ac.uk/

[4] Panero, F. (2024). Lecture 6-8: Neural Networks & Deep Learning [PowerPoint slides]. London School of Economics https://www.lse.ac.uk/