Project - Report

COMP5310

Activity 11 (Monica) - Group 7:

Unikey	SID
rzhu0287	530551405
jfen0380	490025530
kxio4099	490215441

Setup

1.1Topic and research question

This study explores the challenge of forecasting whether a reservation will be made by online or offline methods, using a wide range of data characteristics. The classification of online and offline bookings is highly significant in the hospitality business, as it reveals diverse consumer habits and preferences that have a direct impact on marketing strategies, resource allocation, and customer relationship management. Precisely identifying the booking route can offer valuable insights that empower stakeholders at different levels.

By developing effective pricing and inventory strategies based on an understanding of reserving trends, hotel management can maximize room occupancy and boost revenue. Through the customization of campaigns and special offers, marketing teams can effectively target specific customer segments, thereby increasing conversion rates and customer engagement. These predictions can be employed by analysts and data scientists to improve predictive models and identify emerging trends in customer booking patterns.

1.2. Dataset

The dataset includes both numerical (integers and floats) and categorical (objects) types. The categorical data need to be encoded. Some data are not useful and could be dropped, such as the day of the week and booking IDs. If the distribution between 'Online' and 'Offline' in the market segment type. It might bias the model towards the majority class. Features like ‘number of weekend nights’ and ‘number of week nights’ might be highly correlated with the total stay or with the ‘average price’. High correlation can lead to multicollinearity issues in some models, although logistic regression is generally robust to this. Extreme values in features like average price or lead time can skew the model's understanding and predictions. These need to be handled appropriately, either by normalization or by applying robust methods.

1.3. Modelling agreements

The criteria we chose are F1 and recall. When evaluating the performance of a classification model, F1 score and recall are two such metrics that are often used in machine learning. Recall is defined as a measure of the proportion of actual positives that are correctly identified. The F1 score is defined as the harmonic mean of precision and recall. It conveys the balance between precision and recall through a number. Essentially, it's calculated as 2 * (Precision * Recall) / (Precision + Recall).(Chicco & Jurman, 2020)

Among the research questions, the scenario of predicting the hotel's online and offline booking types is particularly suitable for using the two evaluation indicators of recall (Recall) and F1 score (F1 Score). Recall reduces the loss of important reservations and ensures that the prediction model misses as few as possible For a small but important number of offline bookings, recall rate will be a key metric. For example, if offline bookings may involve large groups or VIP customers, these bookings may have a significant impact on the hotel's revenue and operations. In this case, high recall means that the model is able to capture these key reservations. The F1 score can comprehensively reflect the model's ability to predict accuracy and completeness in handling actual business scenarios. This is particularly important for hotel managers, who need to ensure that all booking types are processed efficiently while avoiding incorrect bookings.

Unikey: kxio4099 model: decision tree

1. Predictive model

1.1. Model Description:

The predictive model I use is a decision tree classifier, which essentially assumes that the data set can be divided into segments that are homogeneous to the target variable. It assumes that features contribute independently to the outcome. Assume that simple decision rules inferred from training data generalize well to predictions. If there are interactions between features that significantly affect the target variable, the independence assumption may not hold.(Rokach & Maimon, 2008) According to the results, this problem does not exist in this data set, so it does not affect the effectiveness. The advantage of this model is that it is easy to interpret and understand, and can handle both numerical and categorical data. But its limitation is that it is prone to overfitting, especially for complex trees. It is also sensitive to noisy data and may create biased trees if certain classes dominate. The dataset we are using includes both categorical(most variables of this dataset) and continuous variables （like "average price"）, and the goal involves classification(predict 'Offline' or 'Online'), so decision trees are appropriate as they can naturally handle this mixture and model non-linear relationships.(Rokach & Maimon, 2008)

1.2. Model algorithm:

The algorithmic principle of this model is that the decision tree uses feature values to divide the data into subsets, repeat this process recursively to form a tree structure with nodes and leaves, and then select the splits to maximize the homogeneity of the subsets based on the target variable. Some of the hyperparameters include: max_depth: which controls the maximum depth of the tree.

min_samples_split: The minimum number of samples a node must have before splitting.

min_samples_leaf: The minimum number of samples a leaf node must have.

Potential changes In addition to limiting the maximum depth of the tree, there are various strategies for pruning decision trees, such as cost-complexity pruning, which also takes into account the size of the tree and the reduction of splitting impurities. (scikit-learn.org) The algorithm execution starts at the root of the tree and evaluates the rules for splitting the data. Choose the split that produces the most homogeneous child nodes. Continue the recursion until stopping criteria are met (e.g., maximum depth or minimum samples per leaf reached) (Shah, 2018). The appendix for specific pseudocode. For the latest developments or adjustments related to my application. I performed Pruning on the model to control model complexity to avoid overfitting and then Hyperparameter Tuning to obtain optimal performance parameters to enhance the performance of the decision tree.

1.3. Model development:

Data preprocessing includes feature scaling, using min-max scaling to normalize the "average price" to normalize its range. Feature encoding, which maps categorical features (such as “room type” and “type of meal”, etc.) to numerical values. and feature engineering, selecting the importance of features bigger than 0.05 after first training and removing irrelevant or redundant features.

For the division of data into training, validation, and test sets. First, 20% of the data was divided as the test set, and 25% of the remaining 80% of the data was divided as the verification set. The training set accounted for 60% to train the model to learn the pattern of the data, and the test set accounted for 20% to test the model. The validation set 20% is used to tune the hyperparameters of the model. (Lee & Hamilton el)

Hyperparameter selection

max_depth: Controls the maximum depth of the tree to prevent model overfitting. Deeper trees may learn specific noise and details of the data. The optimal tree depth can be found empirically through cross-validation using different max_depth values.

min_samples_split: Determines the minimum number of samples a node must have before splitting. This parameter can help avoid overfitting the model on the training data. Higher values bias the model toward simpler decision trees. Empirical validation can be performed by varying this parameter and observing the model's performance on the validation set.

min_samples_leaf: specifies the minimum number of samples that leaf nodes must have, which helps to further reduce the risk of overfitting of the model. Increasing the value of this parameter can increase the generalization ability of the decision tree. (scikit-learn.org)

Empirical validation evaluates the performance of the model by adjusting parameters and using validation set data. During hyperparameter tuning, use GridSearchCV to explore various combinations of hyperparameters to find the best performing model. Use cross-validation to evaluate the performance stability of the model on different subsets of the data set. The final choice is {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2}

Python functions are

DecisionTreeClassifier(): Initialize the decision tree model. random_state=0 ensures repeatability.

train_test_split(): Split the data into a training set and a test set. test_size=0.2 and random_state=0 ensure split ratio and repeatability.

fit(): train the model on the training data.

Predict(): Generate predictions based on test data. precision_score(): Calculates the accuracy of the prediction.

classification_report(): Provides a comprehensive classification indicator report.

GridSearchCV(): Optimizes model parameters through an exhaustive search for specified parameter values. (scikit-learn.org)

2. Model Evaluation and Optimization

2.1. Model Evaluation

measures of success

Recall: Category 0 represents 'Online' and Category 1 represents 'Offline'. For category 0, the recall rate is 0.93; for category 1, the recall rate is 0.87. This indicates that the model has good coverage for predicting Online categories.

F1-Score: For category 0 and category 1, the F1 scores are 0.94 and 0.84 respectively. The F1 score is the harmonic average of precision and recall, providing a comprehensive performance evaluation. (Chicco & Jurman, 2020).

In general, the predicted categories are unbalanced, and the number of samples in category 0 in the data set is significantly more than that of category 1. This may cause the model to overfit the majority category and affect the prediction ability of the minority category. Although the overall accuracy is high, the prediction precision and recall for a few classes are low.

Noise and outliers in the data set may cause the model to learn incorrect or overly complex rules, but the data set before training the final model has been processed very cleanly, so this is one of the reasons for obtaining this good result. The interpretability of this decision tree model is good, but the visualization of this decision tree found that the picture display was not good, so this method was not used to evaluate the interpretability. According to the background of the research problem and data set characteristics, the study hotel reservation is online It is still offline. In a practical sense, online booking is convenient but difficult to communicate. Offline booking is more convenient for communication to meet your own needs. Therefore, the prediction booking method of type of meal, room type, average price, and special requests can reflect the guests very well. The communication needs are used to predict whether the hotel reservation will be online or offline, so the accuracy of the model is high.

The meaning of evaluation metrics: These metrics reflect the performance of the model on specific tasks and help us understand the strengths and weaknesses of the model in different categories. Potential areas for improvement include techniques to oversample minority classes or undersample majority classes, or adjust the weights of classes during model training. Input features can also be further analyzed and optimized, which may include introducing new features or removing noisy features. This can help improve the model's performance on minority classes. The best implementation is to continue adjusting the hyperparameters of the model, which may make the model perform better, but this operation will take more time.

2.2. Model Optimization

I performed model optimization using grid search, which uses different combinations of hyperparameters to optimize model performance. Specifically, set several sets of parameters including max_depth, min_samples_split, and min_samples_leaf for training to obtain the highest score to find the best settings. The results of GridSearchCV show that the best parameter combination is: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2}, thus improving the accuracy and stability of the model. The entire model optimization process is to first train the model directly, then conduct importance analysis to retain important feature values （'number of week nights', 'type of meal', 'room type', 'average price', 'special requests', 'booking status'） and train the model again, then Delete feature rows with small data volume like 'type of meal' only contains very little data (3, 4), then do the third training, and then add more parameters increase model complexity to train the model, and finally select the best parameters through Hyperparameter Tuning to train the best model.

Unikey: jfen0380 model: Random Forest

1.1 Model Description: Random Forest

Technique Description
Random forest is an ensemble learning method that constructs multiple decision trees during training and outputs the majority class or average prediction of individual decision trees. It operates by building a large number of decision trees during training and outputting the majority class or average prediction of individual decision trees.
Assumptions
Random Forest makes several assumptions:

1. Independence between trees: The algorithm operates under the assumption that all trees in the forest possess independent characteristics. This is accomplished through the implementation of bootstrapped datasets and random feature selection.
2. Large number of trees: The performance generally improves with an increase in the number of trees, though at a cost of increased computational complexity.
3. Homogeneity of errors: The assumption is made that the errors are distributed uniformly throughout the dataset, although this may not hold true for data that contains clustered or grouped structures.

Strengths:
Robust to overfitting: In comparison to individual decision trees, it is less susceptible to overfitting as a result of the aggregation of multiple trees.
Handles large data sets with higher dimensionality: Can manage thousands of input variables without variable deletion.
Versatility: Effective for both classification and regression tasks.
Limitations:
Complexity: More complex than decision trees, leading to higher computational costs.
Model interpretability: Less interpretable than decision trees because of its complexity.

1.2 Model Algorithm: Random Forest

Underlying Principles
Random Forest, a widely-used ensemble learning algorithm, combines multiple decision trees to enhance overall prediction performance and reduce overfitting risks. Its core principles include bagging and feature randomness. (IBM n.d.)

Bagging:
Each decision tree within the forest is constructed using a bootstrapped sample. This sample is a subset of training data selected through random sampling with replacement. This ensures the uniqueness of each tree and captures different aspects of the data.
Feature Randomness:
During the construction of trees, the split selection for splitting nodes is not solely based on the best split among all features. Instead, the selection is made from a randomly chosen subset of features. This introduces more diversity among trees, enhancing the model's generalization ability.

Algorithm Execution
The step-by-step execution of the Random Forest algorithm is as follows:

1. Generate N different bootstrapped training samples from the original dataset.
2. For each bootstrapped sample, grow an unpruned decision tree. At each node, randomly select m features out of all M features, and then choose the best feature and split point among these m features to split the node into child nodes. Repeat step 2 until N trees are generated. Repeat this process until all N trees are generated.

3. For classification tasks, prediction is made through majority voting, i.e., choosing the most frequent output class among the N trees. For regression tasks, the prediction is the average of the outputs of the N trees.

For detailed pseudo-code and further technical specifics, please refer to the pseudo-code provided in the appendix (Guo et al. 2019) .

Hyperparameters

Number of trees (n_estimators): More trees generally improve the performance but also increase computational load.

Maximum number of features (max_features): Determines the number of features to consider when looking for the best split.

Maximum depth of the tree (max_depth): Controls the maximum depth of each tree.

Minimum samples split: The minimum number of samples required to split an internal node.

Variations and Advancements

Random Forest continues to evolve with exciting advancements. Extreme Random Trees introduce greater randomization in split selection, frequently leading to quicker training. Additionally, feature-importance driven Random Forests refine feature selection by adjusting the probability of choosing a feature based on its importance score, leading to even more robust models.

Recent Advancements

Recent advancements have focused on improving the efficiency and scalability of the Random Forest algorithm. This includes parallel tree building and integration with big data platforms like Hadoop and Spark. Additionally, efforts to improve interpretability with techniques such as SHAP (SHapley Additive exPlanations) values have gained popularity (Sahlaoui et al. 2021) .

1.3 Model Development: Building the Predictive Model

Data Preprocessing

Data preprocessing plays a pivotal role in constructing a Random Forest model, significantly boosting its performance. Feature scaling addresses features with varying units and scales, often employing normalization or standardization techniques to expedite model convergence. Additionally, dimensionality reduction simplifies the model and reduces training time, particularly when dealing with high-dimensional data. Furthermore, feature engineering enhances the model's predictive power by creating new features or modifying existing ones to uncover hidden relationships within the data.

Division of Data

To ensure the model's generalizability, the data is divided into training, validation, and test sets. The training set is used to build the model and is typically the largest subset, while the validation set helps fine-tune hyperparameters and configuration decisions to prevent overfitting. The test set, completely independent from the training process, is used to evaluate the model's performance.

Strategies

Employing strategies such as temporal validation ensures the model is tested on future, unseen data if the dataset includes a time component. Stratified sampling is advantageous for handling imbalanced datasets by maintaining a consistent proportion of samples for each class across different splits.

Model Implementation and Validation

When utilizing Python's **sklearn**.ensemble's **RandomForestClassifier** or **RandomForestRegressor**, it is common to alter important parameters such as **n_estimators**, **max_features**, and **max_depth**. Theoretical basis for employing Random Forest lies in its capacity to decrease variance through bagging, hence improving the resilience and generalizability of the model on unfamiliar data. Empirical validation is conducted by subjecting the model to testing with different hyperparameters via cross-validation. Metrics such as accuracy, F1-score, and ROC-AUC are employed to evaluate performance and verify the model's efficacy.

2.1 Model Evaluation

For this segment, a comprehensive evaluation of the Random Forest model's performance has been conducted using accuracy, precision, recall, and F1-score as the primary metrics. These metrics are interpreted in the context of the dataset characteristics and the specific research question posed.

Results Interpretation

The Random Forest model demonstrates impressive performance metrics, achieving an accuracy of 93.45%, which indicates a high rate of overall correct predictions. The precision rates for the two classes highlight the model's effectiveness, with 89% accuracy for Offline predictions and an even higher 96% for Online predictions. The recall rates further reinforce this, showing that the model correctly identified 92% of all actual Offline instances and 94% of Online instances. The balanced measure of F1-score, which harmonizes precision and recall, stands at 0.91 for Offline and 0.95 for Online, reflecting strong and consistent performance across both classes. These results illustrate the model’s robust capability in accurately classifying different booking types.

The Precision-Recall curve added to the Random Forest model's evaluation demonstrates a robust balance between precision and recall, maintaining high precision well into higher recall levels, only dropping as recall approaches 1.0. This indicates the model's strong capability in managing class imbalances while maintaining accuracy across a wide range of operational thresholds, further emphasizing its effectiveness in classifying booking types.

Dataset Characteristics and Research Question Context

The dataset exhibits a class imbalance with a higher number of samples for Online bookings compared to Offline, which may contribute to the enhanced performance metrics observed for the Online category. Despite this imbalance, the model demonstrates robustness against potential noise within the dataset, as evidenced by the high precision and recall scores. However, due to the ensemble nature of Random Forest models, they are generally less interpretable. Nonetheless, analyzing feature importance metrics can provide valuable insights into which variables are most influential in driving the predictions, offering some level of interpretability and understanding of the model’s decision-making process.

Implications of Evaluation Metrics and Areas for Improvement

The model's exceptional performance across all measures indicates that it is very suitable for the prediction purpose. Nevertheless, the little disparity in performance among classes suggests the need for additional optimization, namely in addressing class imbalance. Possible areas for enhancement involve exploring methods to address class imbalance more efficiently, such as modifying class weights within the model or utilizing resampling techniques. Additionally, the process of further improving the model might include fine-tuning hyperparameters through techniques like grid search. This would allow for the exploration of various parameter configurations to determine if they can boost the performance of the model.

2.2 Model optimization

To optimize the performance of the Random Forest model, advanced techniques such as hyperparameter tuning were employed using grid search. This involved testing combinations of **n_estimators**, **max_depth**, and **min_samples_split** across 36 different scenarios, totaling 180 fits. The process identified that the optimal parameters for maximizing accuracy were 200 trees (**n_estimators**), no limit on tree depth (**max_depth**), and a minimum of 2 samples required to split an internal node (**min_samples_split**). This configuration improved the model's accuracy to 93.43%, demonstrating robust precision and recall across the classes, with particularly strong performance in identifying the Online bookings. This approach effectively addresses potential overfitting and enhances the model's general applicability to unseen data. The accompanying Precision-Recall Curve, which maintains high precision across most recall levels, further illustrates the model's capability in handling class imbalances and ensuring accuracy in diverse operational scenarios.

SID:530551405 Unikey: rzhu0287

1. Predictive Model: Logistic regression

1.1 Model Description

The model I use is logistic regression. The logistic regression usually fits the binary classification problem. Typically, if the predicted probability is greater than 0.5, the data is classified into the positive class; otherwise, it is classified into the negative class. The predicted labels are two classes which are "Online" and "Offline". These labels could be encoded to numerical labels such as 0 and 1.

1.1.1 Assumptions of linear regression

The target data is binary. The observations are independent. No Multicollinearity of the training dataset. There are no extreme outliers. There is a linear relationship between the training dataset and the testing dataset. The dataset is large enough.

1.1.2 Strengthens and Limitations:

Strengthens:

- Multicollinearity: The data has attributes for describing the features of visitors to the hotel. The attributes do not correspond with each other by their meaning. On the other hand, the correlations have been shown from the previous report stage 1, heatmap.
- Large sample size: The adequate variables in the data set. Logistic regression requires a large number of observations.

Limitations:

- Linearity: It is not sure that there is linearity between the dependent variable and the independent variables.
- Outliers: There are clear outliers for several features: "Number of weeknights", "number of children.", "lead time", and"average price" from boxplot. These outliers might effect the performance of logistic regression since it is sensitive to the outliers.

Justifications:

According to those assumptions of the regular logistic regression, it should be fine used for the current hotel booking data set.

1.2 Model Algorithm

1.2.1 Underlying principles of logistic regression:

Logistic regression is a classification model, that particularly performed well on binary class. The sigmoid function:

$$\sigma(z)= \frac{1}{1+e^{-z}} \text{ ,where } z=w0+w1x1+w2x2+...+wnxn$$

w indicates the weights corresponding to the features *x*1, *x*2, …,*xn, respectfully.*

1.2.2 The logistic function coodinate any real-valued number to the range [0, 1], so they will be able to transit for representing probabilities.

1.2.3 Hyperparameters:

- One Hot Encoding: The method converts values into categorical columns. It signs a new binary column for each category of the variable. The best suitable case is that the nominal data do not have any relationship between the categories which leads models to perform better by treating each category as an independent feature. It is usually suitable for logistic regression. However, it has disadvantages that the method will increase the dimensions which might lead to model complexity.
- Label Encoding: The Label encoding converts each category into an integer value. It is usually used for conducting some labels that have some relationship. For example, online or offline when they are target features.
- Solvers: Solvers are optimizations in the scikit learn for logistic regression which can minimize the cost function by using optimized gradient decent methods.
- Penalties: The default penalty we set for this problem is set as L2. The typical L1 and L2 impose penalties to the coefficients so the model could be generalized and prevent models' overfitting. L1 indicates Lasso regression. It generalized the less important features' coefficient to true zero. It could be useful if there are a lot of features. One case is that the useful features are much less than the total.

Another case is the number of features is much more than the sample size. L2 indicates Ridge regression. It generalized the less important features' coefficient close to zero. In this way, the model is less complex but keeps useful information.

- Maximum Iterations: There is a maximum number of iterations for the grid search taken.
- Random_state: A fixed number signed to random_state guarantees that the same data splits, parameter initializations, or random selections happen in the same way each time when running the code.

1.2.4 Algorithm Execution: (Appendix)

1.2.5 Discussion on recent advancements or adaptations:

The recent logistic regression can use updated techniques, such as scalability, and optimizing solvers for reducing the loss, which let the logistic regression be used on more types of datasets. In the dataset chosen, the experiment should be taken for comparison of which parameters or which combination of parameters will work the best.

Besides, the logistic regression can be integrated into some neural network architectures, enabling end-to-end learning and improved predictive performance. (reference)

1.3 Model Development

1.3.1 Data Preprocessing:

a. Use One Hot Encoding method to conduct only categorical data into numerical data in the sample without the target feature. The method fits the data in the sample well, because of the multicollinearity of the logistic regression.
b. Label Encoding: The target "online" and "offline" is encoded to label as 0 and 1. The features are inverse of each other which means they have inverse relationships. The label Encoding transforms the target features and will not increase the computing complexity.
c. Feature Scaling: Use StandardScaler after the categorical data in the sample that has been one hot encoding.
d. Data split: Data is divided into training set, validation set, and testing sets to ensure the model's ability to generalize to unseen data. The data is split into 60% for training, 20% for validation, and 20% for testing. It is a common procedure if the data is split to 80% and 20% according to Pareto principle(Grosfeld-Nir, et al.). The validation set can be used to tune parameters and select the best model without using the test set, which should only be used to assess the model's final performance.

1.3.2 Python Functions:

- LogisticRegression(): Builds the logistic regression model. The insert parameters are max_iter"=1000, solver="sag", random_state=43
  - The iterations are 1000 for each search.
  - The default solver for this problem is set as "sag". "sag" stands for Stochastic Average Gradient descent (SAG). It stores and updates a simple average of the most recent gradients for each training example. The method is good at optimizing the large dataset. However, the best parameter might not be SAG. The best parameter should be decided by the output of the grid search.
  - The random_stats is 43. It ensures that each output will have the same result.

2. Model Evaluation and Optimization

2.1 Model Evaluation

2.1.1 Confusion matrix

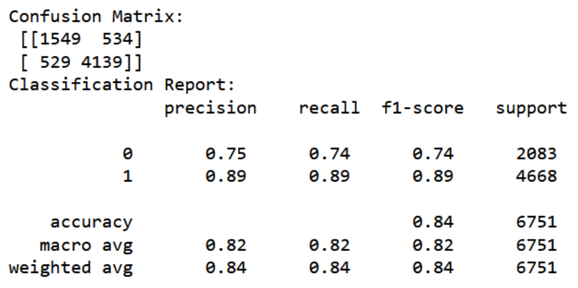
The confusion matrix is used to evaluate the performance of the fitting model. The values of True Positive (TP), True Negative (TN), Positive (FP), False Negative (FN) in the graph could clearly explain the performance of the model. The label 0 is confused by the classification to the label 1 is 534/(1549+534) which is around 25.6%. The label 1 is confused by the classification to the label 0 is 529/(529+4139) which is 11.3%. The precision and recall are almost the same so the overfitting seems not happen.

2.1.2 Classification report

Classification report is used to evaluate model performance, focusing on accuracy, precision, recall, and F1-score. The precision is 0.75 for class 0 and 0.89 for class 1. The Precision score and recall values are almost the same. It seems that overfishing does not happen. $F1=2\times(\text{precision}+\text{recall})/(\text{precision}\times\text{recall})$ which shows the balance of the precision and recall values. Overall, the logistic regression has better performance on class 1.

2.1.3 Receiver Operating Characteristic curve(ROC curve)

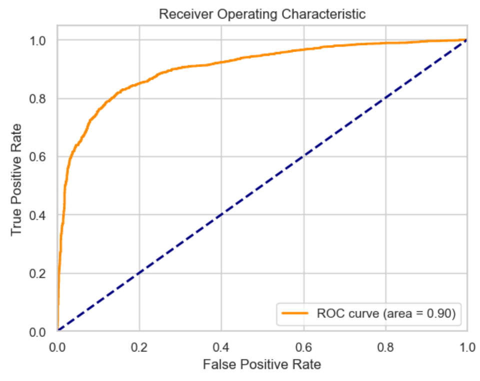
The AUC for your ROC curve is 0.90. This is a high value, suggesting that the model has a good measure of separability. An AUC of 0.90 means there is a 90% chance that the model will be able to distinguish between positive class and negative class.



```
Confusion Matrix:
 [[1549  534]
 [ 529 4139]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.74      0.74      2083
           1       0.89      0.89      0.89      4668

    accuracy                           0.84      6751
   macro avg       0.82      0.82      0.82      6751
weighted avg       0.84      0.84      0.84      6751
```



Model Optimization

2.2.1 GridSearchCV(): The grid search is used to find the highest accuracy and coordinates the best parameters. The “lbfgs”,”newton-cg”,”sag”,”saga” is applied as the parameters tuning for the grid search. The paneities of the parameters are L1 and L2. The strength of the “c” are the list from 0.01 to 100.

- Solvers: In the grid search of the model: Solvers include “lbfgs”, “newton-cg”, “sag”,”saga” will be applied. The “lbfgs”, “newton-cg”, “sag” supports only L2. In the grid search, if the solver does not support the penalty, the round will be directly skipped and would not affect the whole grid search approach.
    - “lbfgs”: Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm. The L-BFGS uses a limited amount of computer memory. It's good for small to medium-sized datasets.
    - “newton-cg”: Newton's method for optimizing, also known as the Newton conjugate gradient method. It is speedy for small- to medium-sized datasets but requires computing the second derivatives, so it is much more intensive-computationally.
    - "sag": Stochastic Average Gradient descent. Only a random subset of the full dataset is used to compute the gradients at each iteration. It's efficient for large datasets and supports only L2 regularization.
    - “saga”: Stochastic Average Gradient Augmented that supports both L1 and L2 regularization,
- Penalties: L1, L2 will be applied. The discuss of the type of penalties is covered in the Model Algorithms.
- “C” represents the inverse of regularization strength. The smaller value, the stronger the regulations. For example: The value of 0.01 might not capture all the patterns in the data, potentially leading to higher bias but lower variance. Inversely, the 100 will capture the data probably including noise which might lead to over fitting.
-

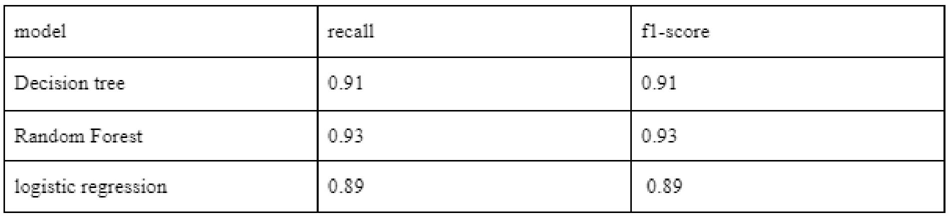
2.2.2 Outliers: The outliers has been removed from the detect of the boxplot. The codes then could been add to the front to achieve better performance. It can been see that the precision score increase from 88 to 89. The method to remove outliers is : velues below $Q1-1.5\times IQR$, will be set to $Q1-1.5\times IQR$, *values above* $Q3+1.5\times IQR$ will be set to $Q3+1.5\times IQR Q3+1.5\times IQR$.

Discussion

One of the greatest advantages of decision trees is their high interpretability. The model's decision-making process can be visually displayed through a tree diagram. and can handle non-linear relationships. For limitation, decision trees are very prone to overfitting, especially when the tree is very deep or does not have proper pruning. Decision trees are very sensitive to small changes in the data and make the prediction results of a single decision tree potentially unstable.

The Random Forest model is renowned for its robustness, as it is less prone to overfitting in comparison to individual decision trees. Furthermore, it has remarkable scalability, effectively handling extensive datasets filled with numerous variables without requiring the elimination of any. Nevertheless, it has several restrictions; its intricacy surpasses that of decision trees, which could potentially escalate computational requirements and prolong processing time. Furthermore, the intricate nature of the model can reduce its interpretability.

We evaluated the Logistic Regression algorithm for our dataset. Strengths include a lack of multicollinearity and a large sample size of 23,626 observations, which supports robust model training. However, the model faces challenges due to the dataset's high dimensionality and uncertainty in the linearity between variables. Additionally, significant outliers in features such as "number of weeknights" could negatively impact the model's performance, as Logistic Regression is sensitive to outliers.



model	recall	f1-score
Decision tree	0.91	0.91
Random Forest	0.93	0.93
logistic regression	0.89	0.89

Quantitative comparison: Recall: Random forest shows the highest recall, which is 0.93. This shows that among the three models, Random Forest is the most effective in identifying cases below the scheduled line. The recall rate of the decision tree follows closely at 0.91, and its performance is also very good. Logistic regression has a relatively low recall rate of 0.89, which may mean it is the worst at avoiding missing positive classes.

F1-Score: Random Forest also performed best on F1-score of 0.93, which reflects that it achieves the best balance between precision and recall. The F1 score of the decision tree is 0.91, which is a good performance. Logistic regression has an F1 score of 0.89, which is the lowest, indicating that it is the weakest at balancing precision and recall.

Qualitative comparison: Random forests are generally more complex than decision trees and may require more computing resources, but therefore provide more stable performance. Decision trees are relatively simple and intuitive, easy to understand and explain, but may face the risk of overfitting. Logistic regression provides good interpretability, but may not be as effective as tree models when dealing with complex or nonlinear relationships.

Wider implications for the research question: In predicting hotel reservations, high-recall model random forest can help ensure that almost all important offline reservations are identified, which can ensure customer satisfaction and optimize hotel revenue. A model with a high F1 score ensures that while accurately identifying booking types, it misses as few important bookings as possible, helping to maintain service quality and customer relationships.

Conclusion

Our thorough assessment of decision trees, Random Forest, and Logistic Regression models for predicting hotel bookings concludes that there are clear benefits and difficulties associated with each. The Random Forest model demonstrates great effectiveness by achieving the maximum recall and F1-score of 0.93, suggesting its exceptional performance in accurately detecting and balancing booking kinds. Although decision trees are highly interpretable and user-friendly, they are susceptible to overfitting and exhibit lower stability. Although Logistic Regression is known for its interpretability, it is not effective in dealing with complicated datasets and non-linear relationships. Considering the crucial importance of precisely identifying bookings to maximize income and customer happiness, it is advisable to use the Random Forest model because of its strong accuracy. Potential future investigations may involve optimizing tree-based models, and improving data collection techniques to minimize outliers.

Reference

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 6–6. https://doi.org/10.1186/s12864-019-6413-7

Decision trees. Available at: https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/ (Accessed: 06 May 2024).

Grosfeld-Nir, A., Ronen, B., & Kozlovsky, N. (2007). The Pareto managerial principle: when does it apply? *International Journal of Production Research*, *45*(10), 2317–2325. https://doi.org/10.1080/00207540600818203

Guo, H., Nguyen, H., Vu, D.-A., & Bui, X.-N. (2019). Forecasting mining capital cost for open-pit mining projects based on artificial neural network approach. *Resources Policy, 74*. https://doi.org/10.1016/j.resourpol.2019.101474

IBM. (n.d.). What is Random forest? Retrieved May 7, 2024, from https://www.ibm.com/topics/random-forest

Lee, S. B., Gui, X., Manquen, M., & Hamilton, E. R. (n.d.). Use of Training, Validation, and Test Sets for Developing Automated Classifiers in Quantitative Ethnography. In *Advances in Quantitative Ethnography* (pp. 117–127). Springer International Publishing. https://doi.org/10.1007/978-3-030-33232-7_10

Pseudocode of logistic regression. (n.d.). ResearchGate. https://www.researchgate.net/figure/Pseudocode-of-logistic-regression_fig15_343120305

Rokach, L., & Maimon, O. Z. (2008). *Data mining with decision trees theory and applications*. World Scientific.

Shah, C. (2018). *Classification : theory - decision trees and random forest.* Chirag Shah.

Sahlaoui, H., Alaoui, E. A. A., Nayyar, A., Agoujil, S., & Jaber, M. M. (2021). Predicting and interpreting student performance using ensemble models and Shapley additive explanations. IEEE Access, 9, 152688-152703. https://doi.org/10.1109/ACCESS.2021.3124270

Sklearn.tree.decisiontreeclassifier (no date) scikit. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (Accessed: 07 May 2024).

Scikit-learn. (2014). *sklearn.linear_model.LogisticRegression — scikit-learn 0.21.2 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

SciKit Learn. (2019). *sklearn.model_selection.GridSearchCV — scikit-learn 0.22 Documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

\

Appendix

Pseudo code for the Decision Tree algorithm

```
Function BuildDecisionTree(Dataset):
     If all records in the dataset have the same target variable value:
        -> Return a leaf node with the label of that common value
    If there are no features left to split on:
        -> Return a leaf node with the label of the most common target variable value in the dataset
    Select the best splitting feature:
        For each feature in the dataset:
            For each possible value of the feature:
                Split the dataset based on that feature value
                Evaluate the effectiveness of the split (typically using Gini impurity or information gain)
            Record the best split effectiveness for that feature
        -> Choose the feature with the best split effectiveness for division
    Using the selected feature and corresponding best split point, divide the dataset into subsets
    Recursively call BuildDecisionTree function on each subset
    Create a decision node linked to the subtrees generated in the steps above
    Return the decision node
```

(cs.cmu.edu)

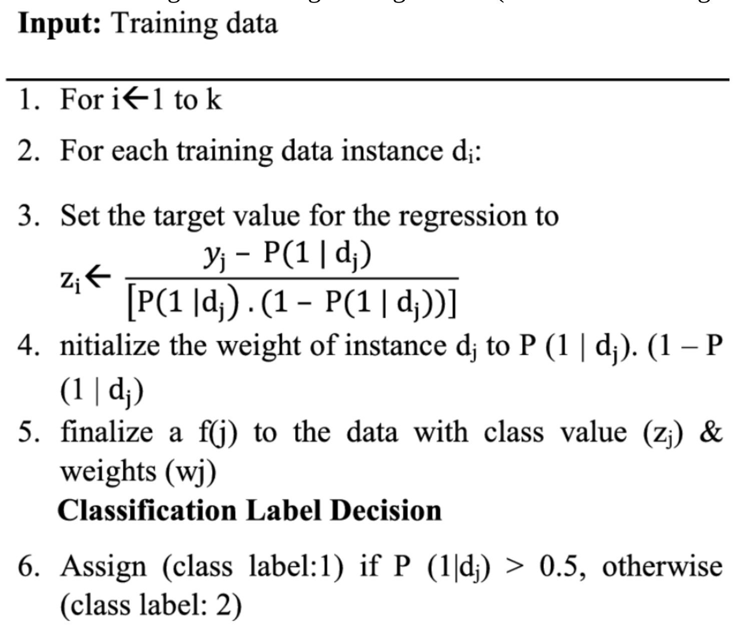
Pseudo code for the Random Forest algorithm

```
To generate c classifiers:
for i = 1 to c do
  Randomly sample the training data D with replacement to produce D_i
  Create a root node, N, containing D_i
  Call BuildTree(N)
end for
BuildTree(N):
```

```
if N contains instances of only one class then
  return
else
  Randomly select x% of the possible splitting features in N
  Select the feature F with the highest information gain to split on
  Create child nodes of N, N_1, ..., N_F, where F has f possible values
  for i = 1 to f do
    Set the contents of N_i to D_i, where D_i is all instances in N that match F_i
    Call BuildTree(N_i)
  end for
end if
```

(Guo et al. 2019)

Pseudo-code algorithm of logistic regression: (Pseudocode of Logistic Regression, n.d.)



```
Input: Training data
1. For i ← 1 to k
2. For each training data instance $d_i$:
3. Set the target value for the regression to
   $z_i \leftarrow \frac{y_j - P(1 \mid d_j)}{[P(1 \mid d_j) \, . \, (1 - P(1 \mid d_j))]}$
4. nitialize the weight of instance $d_j$ to P (1 | $d_j$). (1 – P (1 | $d_j$)
5. finalize a f(j) to the data with class value ($z_j$) & weights (wj)
   Classification Label Decision
6. Assign (class label:1) if P (1|$d_j$) > 0.5, otherwise (class label: 2)
```