EXPERIMENT REPORT

|  |  |
| --- | --- |
| Student Name | Balakumaran Sivanesan |
| Project Name | Assignment 2 - Classification Models: Experiment 1 |
| Date | 28th April 2024 |
| Deliverables | <notebook name: rfc.ipynb>  <model name: clf> |

|  |  |
| --- | --- |
| 1. EXPERIMENT BACKGROUND | |
| Provide information about the problem/project such as the scope, the overall objective, expectations. Lay down the goal of this experiment and what are the insights, answers you want to gain or level of performance you are expecting to reach. | |
| 1.a. Business  Objective | The objective of this project is to create a predictive model for the business that can correctly identify clients based on their car model, auto category, and demographics who are most likely to make another purchase from the company. The project's outcomes will be put to use by the business to improve client retention rates and marketing tactics. The project makes use of GridSearchCV for hyperparameter tuning and the Random Forest classifier method to forecast the chance of customer repurchase based on a variety of factors.    If the model produced findings that were correct, the firm would benefit greatly because it could use the information to identify consumers who are most likely to make repeat purchases and target them with offers and incentives that would increase their loyalty. Revenue would rise as a result, and consumer happiness would rise. However, inaccurate outcomes can result in the misallocation of marketing funds and the loss of possibilities to hold onto important clients, which would reduce customer satisfaction and result in lost revenue. |
| 1.b. Hypothesis | This experiment aims to test the hypothesis that a Random Forest classifier model can reliably forecast a customer's likelihood of making another purchase based on the customer's demographic data, car model, and automobile segment. It provides an answer to the following questions: can the generated model reliably identify consumers who are most likely to make another auto purchase from the company? How well does it perform in terms of forecasting the outcome of a repurchase?  The performance evaluation of the constructed model using many metrics, including F1-score, ROC AUC score, accuracy, precision, and recall, is the insight that this code offers. These measures are used to evaluate the model's generalization performance, prevent overfitting, and evaluate the model's accuracy, consistency, and ability to classify true positives and true negatives.  Additionally, the GridSearchCV is used to tune the model's hyperparameters, which can enhance the model's accuracy and resilience. Analyzing the model's performance with a variety of metrics can reveal the model's advantages and disadvantages as well as point out areas in need of development. All things considered, the code's hypothesis testing is significant for organizations since it sheds light on consumer behavior and aids in the creation of winning marketing campaigns that increase revenue and retention rates. |
| 1.c. Experiment  Objective | The experiment's goal is to create a Random Forest classifier model that can correctly forecast a customer's likelihood of making another purchase based on their car's model, segment, and demographic data. Using the testing set, the model's accuracy, precision, recall, F1-score, and ROC AUC score will be assessed. The results will be compared to the baseline model.  A model with high recall, accuracy, and precision is desired, as is one with a high ROC AUC score that demonstrates a good balance between true positive and false positive rates. The target ROC AUC value is greater than 0.80, and the estimated accuracy objective is approximately 85%.    Possible scenarios resulting from this experiment include:   1. Best case scenario: The constructed model meets or beyond the anticipated goals with high accuracy, precision, recall, F1-score, and ROC AUC score. This would suggest that the model can correctly forecast the chance that a client would make another purchase and that it can be applied to improve the company's marketing tactics, leading to higher rates of customer retention and income. 2. Acceptable scenario: The constructed model outperforms the baseline model while achieving only moderate levels of recall, accuracy, precision, F1-score, and ROC AUC score. This suggests that the model can be used to somewhat enhance the company's marketing strategy and has some predictive value.      1. Unacceptable scenario: The constructed model has little to no predictive potential as evidenced by its poor performance, which includes low recall, accuracy, precision, F1-score, and ROC AUC score. This would imply that alternative techniques would need to be investigated and that the model cannot be utilized to enhance the business's marketing strategies     The experiment's overall goal is to create a predictive model that will help the business enhance its marketing tactics and client retention rates. The experiment's prospective outcomes will also shed light on the model's viability and probable effects. |

|  |  |
| --- | --- |
| 2. EXPERIMENT DETAILS | |
| Elaborate on the approach taken for this experiment. List the different steps/techniques used and explain the rationale for choosing them. | |
| 2.a. Data Preparation | In the given code, the following steps were taken for preparing the data:   1. Remove unnecessary columns from the dataset: The 'ID' and 'age\_band' columns were removed. These columns were eliminated since it was determined they had no bearing on forecasting the chance that a customer would make another purchase. 2. Encode category variables: LabelEncoder, which transforms categorical data into numeric values, was used to encode the 'gender' column. This was required since categorical variables can only be included in the model by encoding them, whereas machine learning algorithms usually operate on numerical data.      1. One-hot encoding of categorical variables: pd.get\_dummies was used to one-hot encode the 'car\_model' and 'car\_segment' columns. This was required since LabelEncoder could not encode these variables because they are category and have numerous values. Categorical data are transformed into several binary variables, each of which represents a potential category value, using one-hot encoding. 2. Divide the data into training and testing sets: Train\_test\_split from sklearn.model\_selection was used to divide the data into training and testing sets. In order to train the model on a subset of the data and assess its performance on another subset, this was done.   Since the supplied data was reasonably clean and didn't require a lot of preprocessing, no extra steps were performed to prepare it. |
| 2.b. Feature  Engineering | No specific feature engineering procedures were followed. On the other hand, the 'car\_model' and 'car\_segment' columns underwent one-hot encoding, which is akin to feature engineering. To transform categorical features into a format that may be utilized as input for machine learning algorithms, one-hot encoding is justified. Here, the one-hot encoded characteristics indicate whether a specific automobile model or car segment is present or absent for a given observation.  One may argue that the removal of the 'age\_band' column from the dataset amounted to feature removal. It's unclear from the provided code why it was done, but it might have been done if the feature was thought to be superfluous or unnecessary for the study.   There are no features that were considered crucial for further research because there were no additional explicit feature engineering or feature removal procedures performed in the code. |
| 2.c. Modelling | The Random Forest classifier is the model utilized in this experiment. Several decision trees are combined in the well-liked ensemble learning technique Random Forest to produce a more reliable and accurate model. It is renowned for its capacity to manage high-dimensional datasets and prevent overfitting and is frequently employed for classification jobs.    In the given code, the hyperparameters tuned using GridSearchCV are as follows: - `n\_estimators`: the number of decision trees in the forest. The values tested were 50, 100, and 150.   * `max\_depth`: the maximum depth of the decision trees. The values tested were None (unlimited), 5, and 10. * `min\_samples\_split`: the minimum number of samples required to split an internal node. The values tested were 2 and 5. * `min\_samples\_leaf`: the minimum number of samples required to be at a leaf node. The values tested were 1 and 2. * `max\_features`: the maximum number of features to consider when looking for the best split. The values tested were 'auto', 'sqrt', and 'log2'.   The goal of selecting these hyperparameters is to identify the set of values that maximizes the model's performance. These hyperparameters were selected for Random Forest classifiers using common values found in the literature and the scikit-learn manual.  No models were chosen not to train since no additional models were trained using the provided code. It's crucial to remember that there are a variety of alternative models, like support vector machines, gradient boosting machines, and neural networks, that could be taken into account for this task. The particulars of the issue and the available data will determine which model is best.  Numerous other factors, including the splitting criterion, the maximum amount of samples for each tree, or the bootstrap sampling approach, could be taken into consideration for tuning in relation to hyperparameters. Investigating how various feature engineering or feature selection strategies affect the model's performance may also be helpful. These are possible areas for additional research in the future. |

|  |  |
| --- | --- |
| 3. EXPERIMENT RESULTS | |
| Analyse in detail the results achieved from this experiment from a technical and business perspective. Not only report performance metrics results but also any interpretation on model features, incorrect results, risks identified. | |
| 3.a. Technical  Performance | The random forest model's performance was assessed using a number of metrics, including ROC AUC score, accuracy, precision, recall, and F1-score.    The model's performance was marginally enhanced by the hyperparameter tweaking done with GridSearchCV. The following were the optimal hyperparameters that GridSearchCV discovered:   * 'max\_depth': 10 * 'max\_features': 'sqrt' * 'min\_samples\_leaf': 1 * 'min\_samples\_split': 2 – * 'n\_estimators': 150   The tuned model achieved the following performance metrics on the test set:    The accuracy and precision of the adjusted model were somewhat higher, but the recall score dropped. This implies that there is still difficulty for the model to accurately identify all of the positive cases.    Examining the confusion matrix and verifying the false negatives (instances where the model projected the client would not repurchase, but they actually did) will help identify the primary underperforming cases and observations. To check whether there are any specific patterns or attributes connected to these situations that could be utilized to enhance the model's functionality, it could be worthwhile to take a closer look at the data.   |  |  |  | | --- | --- | --- | |  | Predicted No | Predicted Yes | | Actual no | 83,836 | 36 | | Actual yes | 274 | 754 |  * 83,836 customers were accurately predicted not to repurchase a car ("No") – these are true negatives. * 754 customers were accurately predicted to repurchase a car ("Yes") – these are true positives. * 36 customers were predicted to repurchase a car ("Yes") but did not – these are false positives. * 276 customers were not predicted to repurchase a car ("No") but they did – these are false negatives.   The class imbalance in the data, where the positive class (i.e., customers who repurchased) is considerably smaller than the negative class (i.e., customers who did not repurchase), and the small size of the dataset could be the primary causes of the model's underperformance. These elements could make it challenging for the model to pick up patterns connected to the positive class, which would lead to poorer recall ratings.  Future research should look into alternative algorithms, including cost-sensitive learning or ensemble approaches, that can manage class imbalance more skillfully. Furthermore, utilizing more sophisticated feature engineering approaches and gathering additional data could enhance the model's performance. |
| 3.b. Business Impact | RandomForestClassifier model's performance saw a marginal improvement, achieving an accuracy of about 99.2% and an F1-score around 82.9% on the test set, indicating that the adjustments made had a slight but positive impact on the model's predictive capabilities.  It seems that the model may reasonably anticipate repurchase behavior when interpreting these results in light of the business purpose. The comparatively low F1-score, however, suggests that there is still opportunity for development, especially with regard to striking a balance between recall and precision. This implies that certain clients may be misclassified by the model as likely to repurchase when in fact they won't, or vice versa.    There are a few distinct scenarios to think about when it comes to the possible effects of inaccurate results on the business. The company may waste money on marketing and retention initiatives that prove to be futile if the model forecasts erroneously that a client is likely to repurchase when they actually won't. On the other side, the firm may lose out on opportunities to keep a valuable customer if the model forecasts inaccurately that a customer won't repurchase when they actually will.  Generally speaking, precise forecasts of repurchase behavior can aid a company in maximizing customer loyalty and income through marketing and retention initiatives. But it's crucial to remember that a model is only one tool in a bigger picture, and other elements like product quality and customer service could also be very important for retaining customers. |
| 3.c. Encountered  Issues | A list of the problems encountered throughout the trials, together with their fixes or workarounds, is provided below:  1. Unbalanced dataset: There were more negative samples than positive samples in the dataset utilized for the studies. The model may perform badly in the positive class as a result of this. To guarantee that both classes were represented in the training and testing sets, stratified sampling was employed during the train-test split. Using methods to balance the dataset, such as oversampling or undersampling, is another option. It is imperative that future studies take into account the impact of data imbalance and select suitable strategies to address it.  2. Selecting features: The dataset had a large number of features, some of which would not have been pertinent to the target variable. To determine which features have the greatest influence on the target variable, feature selection is crucial. In this experiment, the random forest feature importance was used to pick features, while domain knowledge was used to delete some features. Other feature selection methods, such as PCA or Lasso regression, may be employed in further studies.  3. Model selection and hyperparameter tuning: It's critical to select the appropriate models and hyperparameters for the given situation out of the wide range available. A random forest classifier was chosen in this experiment since it is a reliable model that performs well with tabular data. Grid search was used to identify the ideal hyperparameters through hyperparameter tuning. Other models and more sophisticated hyperparameter tuning methods, such as Bayesian optimization, can be investigated in subsequent studies. |

|  |  |
| --- | --- |
| 4. | FUTURE EXPERIMENT |

|  |  |
| --- | --- |
| Reflect on the experiment and highlight the key information/insights you gained from it that are valuable for the overall project objectives from a technical and business perspective. | |
| 4.a. Key Learning | We learned more about the Random Forest Classifier's efficacy in forecasting customer repurchase based on the experiment's results. The model demonstrated effectiveness in identifying customers who are likely to make repeat purchases, as seen by its relatively high accuracy, precision, recall, and F1-score. The outcomes of the hyperparameter tuning also demonstrated that the model's performance can be enhanced by raising the maximum depth of the tree and the number of estimators while lowering the minimum number of samples needed to split a node and the minimum number of samples needed to be at a leaf node.  Nevertheless, the model's performance still has potential for enhancement. A plausible concern is the dataset's class imbalance, which could have impacted the model's precision in forecasting the minority class. Furthermore, the dataset could not have included all relevant variables that could help anticipate client repurchases. To overcome these problems, more research might be conducted. For example, various resampling strategies could be used to address class imbalance, or more data could be gathered to include more informative features.  Based on the knowledge gathered from this experiment, it appears beneficial to continue experimenting with the existing strategy because there is room for the model's performance to be further enhanced. |
| 4.b. Suggestions / Recommendations | Here are some possible tests and subsequent actions based on the project's main goal and the results attained:   1. Make feature engineering better: Enhancing feature engineering has the potential to improve performance because it is a crucial step in creating successful machine learning models. This can be accomplished by employing more advanced feature engineering approaches or by gathering more data. 2. Tuning hyperparameters: A model's performance can be greatly affected by its hyperparameters. Optimizing the hyperparameters of the existing model may improve performance. 3. Ensemble models: To aggregate the predictions of several models and perhaps improve performance, ensemble techniques including bagging, boosting, and stacking should be investigated. 4. Implement the model in a production environment: The model would be implemented in a production environment if it produced the desired results for the company. This would entail putting up monitoring to make sure the model is operating as intended, integrating the model with current systems, and developing a pipeline for preprocessing fresh data 5. Gather more information: If the size or breadth of the existing dataset is constrained, gathering additional information may help the model function better.   The particulars of the problem and the data greatly influence the possible benefits of each of these procedures. On the other hand, considerable performance gains may usually be achieved by utilizing ensemble techniques, optimizing hyperparameters, and investigating alternative machine learning models.  Overall, considering the outcomes, it appears that continuing to experiment with the existing strategy is a sensible course of action. |