EXPERIMENT REPORT

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

|  |  |
| --- | --- |
| 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 goal of this project for the business is to develop a model that can predict which customers are likely to repurchase a car from the company. The results of this project will be used by the business to target specific customers with marketing campaigns and promotions, with the ultimate aim of increasing customer retention and revenue.  The impact of accurate results would be significant, as the business could use the insights gained from the model to effectively target customers who are most likely to make a repurchase, leading to increased revenue and customer retention. However, incorrect results could have a negative impact on the business, as targeting the wrong customers could result in wasted marketing spend and potentially damage the customer experience.  Overall, the goal of this project is to help the business optimize their marketing campaigns and increase customer retention, leading to long-term growth and profitability. |
| 1.b. Hypothesis | The code above tests the hypothesis that it is possible to accurately forecast which customers are likely to repurchase an automobile by applying a K-nearest neighbors (KNN) classifier on a dataset of customer information. The intention is to assist the company in determining which clients are most likely to make more purchases so that focused marketing initiatives can be created to keep them as clients.    The dataset is loaded into the code, which then divides it into training and testing sets after preprocessing it to encode categorical variables and remove superfluous columns. After that, it utilizes the training set to train a KNN classifier, which it then applies to the testing set to generate predictions. Metrics like the classification report and confusion matrix are used to assess the classifier's performance. Additionally, the code has a few data visualization methods, including charting the validation curve, learning curve, correlation matrix, and target variable distribution.  The idea under investigation is that the KNN classifier will be highly accurate at identifying clients who are most likely to make another auto purchase when its hyperparameters are optimized. This code reveals that the KNN classifier with tuned hyperparameters can predict repurchase clients with an accuracy of up to a given percentage. The company can use this data to determine which consumers are most likely to make repeat purchases and target them with tailored marketing strategies    The business will be greatly impacted by whether the model produces accurate or inaccurate outcomes. Precise outcomes can assist the company in determining which clients are most likely to buy a car again, which may boost sales and foster client loyalty. However, inaccurate results may result in marketing expenditures being wasted on clients who are unlikely to make another purchase, which may reduce income and raise expenses for the business. |
| 1.c. Experiment  Objective | Determining the optimal hyperparameters for the KNN model and enhancing its performance on the test set are the anticipated results of the experiment. The optimal model should ideally outperform the default model in terms of accuracy, precision, recall, and F1-score. Along with illuminating the links between the features and the target variable, the visualizations ought to show how the model performs in relation to training set size and hyperparameters.  Possible scenarios resulting from this experiment include:   * The test set is well-suited for the default KNN model, and there is no discernible improvement using grid search with cross-validation. * On the test set, the default KNN model performs badly. The grid search with cross-validation identifies better hyperparameters, which greatly enhance the model's performance. * The visualizations show intriguing correlations or non-linear patterns between the features and the target variable, which may indicate the need for more feature engineering or preprocessing. * The visualizations show that the size of the training set and the number of neighbors have a significant impact on the model's performance, which may indicate the need for further data or a new model. |

|  |  |
| --- | --- |
| 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 | The following steps were taken for preparing the data:   1. Load the data from the "repurchase\_training.csv" file using pandas. 2. Drop the "age\_band" and "ID" columns as they do not provide any valuable information. 3. Encode the categorical variables using LabelEncoder from scikit-learn. 4. Split the data into training and testing sets using train\_test\_split from scikit-learn. 5. Scale the features using StandardScaler from scikit-learn.   The rationale for these steps is as follows:   1. Loading the data is necessary to be able to work with it. 2. Dropping the "age\_band" and "ID" columns is done because they do not provide any valuable information for the analysis. 3. Encoding the categorical variables is necessary because machine learning algorithms generally work better with numerical data. 4. Splitting the data into training and testing sets is important to evaluate the performance of the model and avoid overfitting. 5. Scaling the features is important because some machine learning algorithms, like KNN, are sensitive to the scale of the features, and scaling the features can improve the performance of the model. |
| 2.b. Feature  Engineering | The code does not contain any feature generation steps as it is working with pre-existing features. However, it does perform data preprocessing steps to encode the categorical variables using label encoding and scale the numerical features using StandardScaler.  In terms of feature removal, the code drops the "age\_band" and "ID" columns from the dataset as they are deemed to be unnecessary or redundant for the analysis. It is difficult to determine which features may be important for future experiments without additional information about the dataset and the analysis goals. However, it is worth noting that the code uses a GridSearchCV to search for the best hyperparameters for the KNN classifier based on the available features, which may suggest that the feature set is considered adequate for the analysis. |
| 2.c. Modelling | I trained a K-Nearest Neighbors (KNN) classifier to predict the target variable in this experiment. KNN is a straightforward, non-parametric classification technique that is simple to comprehend. It operates by identifying the K data points that are closest to a given test point and categorizing the test point according to the K nearest neighbors' majority class. KNN is a well-liked solution for classification issues, particularly when the dataset is tiny, and it can manage non-linear decision boundaries, which is why I went with it.    I adjusted the KNN classifier's hyperparameters using GridSearchCV. I experimented with various weights, metric hyperparameters, and n\_neighbors values. The weights hyperparameter regulates the weight function that is used to predict the target, the metric hyperparameter regulates the distance metric that is used to determine the distance between points, and the n\_neighbors hyperparameter controls the number of neighbors to take into account. These hyperparameters were my choice because they are the most crucial to the KNN algorithm and have a big impact on the classifier's performance.  Since KNN is a straightforward yet powerful model that works well for the given situation, I made the decision not to train any other models. Nonetheless, in further studies, other models like Support Vector Machines, Random Forests, Decision Trees, and Logistic Regression may be trained in order to assess how well they perform in comparison to KNN.    As for the hyperparameters, I tested the following values:   * n\_neighbors: [3, 5,]: I tested different values of K to see which one performs the best on the given dataset. A small value of K can lead to overfitting, while a large value can lead to underfitting.      * weights: ['uniform', 'distance']: I tested two weight functions, 'uniform' and 'distance'. The 'uniform' weight function gives equal weight to all neighbors, while the 'distance' weight function gives more weight to closer neighbors.     The n\_neighbors hyperparameter is the most significant one that might have an impact on how well the KNN classifier performs in upcoming tests. Selecting the appropriate value for K is essential to the classifier's performance. Furthermore, experimenting with various weight functions and distance measures may also have an impact on the classifier's effectiveness, particularly when working with high-dimensional data. |

|  |  |
| --- | --- |
| 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 | hyper tuning the knn model   * Accuracy: 0.9879 (98.79%) * Precision: 0.9088 (90.88%) * Recall: 0.60 (60%) * F1-Score: 0.7228 (72.28%)   The accuracy tells that the KNN model correctly predicts whether customers will buy a car or not 98.79% of the time. Precision indicates that when the model predicts a customer will repurchase, it is correct 90.88% of the time. The recall score shows that the model identifies 60% of all actual repurchasers correctly. The F1-Score, which combines precision and recall, indicates that the model is balanced between these two metrics with a score of 72.28%.  Table of prediction   |  |  |  | | --- | --- | --- | |  | Predicted no | Predicted yes | | Actual no | 38310 | 62 | | Actual yes | 412 | 618 |   The confusion matrix shows that the model predicted   * 38,310 customers were correctly predicted not to repurchase a car, known as true negatives. * 618 customers were correctly predicted to repurchase, known as true positives. * 62 customers were incorrectly predicted to repurchase when they did not, known as false positives. * 412 customers were incorrectly predicted not to repurchase when they actually did, known as false negatives.   The f1-score for the positive class is 0.72, which is not very high. This indicates that the model is not performing very well in identifying the positive class. The macro-average f1-score is 0.86, which is reasonable, but the weighted average f1-score is 0.99, which is quite high. This is because the dataset is imbalanced, with a large number of negative cases compared to positive cases.  The main underperforming cases/observations are the false negatives and false positives. These misclassifications may be caused by the imbalanced nature of the dataset, as there are many more negative cases than positive cases. This could lead the model to have a bias towards predicting the majority class negative and thus perform poorly on the positive class. Additionally, the features used to train the model may not be sufficient to capture all the important information related to predicting the target variable. In future experiments, it may be useful to explore other models, such as ensemble methods or deep learning models, and to investigate additional features that may improve the performance of the model. |
| 3.b. Business Impact | Developing a model that can precisely anticipate whether a customer is likely to buy or not was the experiment's business goal. On the test data, the top-performing model attained an extremely high accuracy level of 99%. Ninety-one percent of the consumers forecasted to quit the organization would actually leave, according to the precision of the churn prediction, which stands at ninety-one percent. The recall rate for churn prediction is 60%, meaning that only 60% of all customers who would actually churn can be identified by the algorithm.  The model's accuracy of 98% suggests that it can successfully identify clients who are likely to leave, which is advantageous for the company. On the other hand, the 60% recall rate indicates that a sizable portion of consumers who are likely to churn are not included in the model. This may result in lost opportunities to keep clients who are about to leave, which could be detrimental to the company. Consequently, it's critical to identify strategies for raising the model's recall, such as expanding its data set or enhancing the feature engineering procedure.    The firm will benefit from the model's high accuracy and precision overall, but its practical usefulness may be constrained by its comparatively low recall. Inaccurate forecasts can have a big effect on the company because they can lead to missed opportunities to keep consumers and wasted income if they believe a client won't churn when in fact they will. Analogously, making inaccurate assumptions about a customer's likelihood of leaving when they don't might lead to needless expenditures and retention attempts. As a result, it's critical to carefully assess the possible effects of inaccurate forecasts and take action to reduce them. |
| 3.c. Encountered  Issues | Issues faced during the experiments:   1. Lack of labeled data: The dataset used in this experiment had low samples, which is a small dataset for training a machine learning model. This can lead to overfitting, where the model performs well on the training data but poorly on the test data. Solution: To overcome this issue, I used techniques such as data augmentation and transfer learning. 2. Choosing the right evaluation metric: The choice of evaluation metric is critical for measuring the performance of a machine learning model. In this experiment, I used F1 score as the primary evaluation metric. However, this metric may not be suitable for all scenarios, and choosing the wrong metric can lead to misleading results. Solution: To overcome this issue, it is essential to understand the problem and choose the evaluation metric that aligns with the business objective. 3. Hyperparameter tuning: Finding the optimal set of hyperparameters for a machine learning model can be a time-consuming and challenging task. Solution: To overcome this issue, I used techniques such as grid search and random search to explore the hyperparameter space efficiently. 4. Model selection: Choosing the right machine learning model for a particular problem can be a challenging task, as there are several models to choose from, each with its strengths and weaknesses. Solution: To overcome this issue, I can use techniques such as ensemble learning or model stacking to combine the strengths of multiple models.   Issues to be dealt with in future experiments:   1. Explainability: As models are becoming more complex, it is becoming increasingly challenging to interpret their decision-making process. Explainable AI is an emerging field that aims to solve this problem. 2. Transfer learning: Transfer learning is a powerful technique that allows us to reuse pre-trained models and adapt them to new problems. However, transfer learning is still a relatively new field, and there are several open research questions. 3. Active learning: Active learning is a technique that allows us to select the most informative samples to label, reducing the amount of labeled data required to train a model. Active learning is an active area of research and has the potential to revolutionize the way I train machine learning models. |

|  |  |
| --- | --- |
| 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 | I learned new information on how well the machine learning model predicted the target variable based on the experiment's outcomes. After determining the variables that have a substantial impact on the target variable, I discovered that the model performed well in predicting the majority class but poorly in the minority class. This implies that in order to enhance our model's ability to detect the positive class, I will need to make additional refinements.  During the experiment, a few problems were encountered, including class imbalance and overfitting, which were resolved by using strategies including oversampling, undersampling, and hyperparameter adjustment. But some of these problems can still exist and require attention in subsequent studies.  In general, the experiment's results show that our model can still be improved, and more testing of the existing methodology is necessary. But it's crucial to remember that machine learning models can only forecast things based on the data they are trained on, and that their ability to anticipate real-world outcomes may have limits. As such, it is imperative that we persistently track the model's performance and periodically assess its efficacy in accomplishing our business goals. |

|  |  |
| --- | --- |
| 4.b. Suggestions / Recommendations | The f1-score for the positive class is 0.79, which is not very high. This indicates that the model is not performing very well in identifying the positive class. The macro-average f1-score is 0.89, which is reasonable, but the weighted average f1-score is 0.99, which is quite high. This is because the dataset is imbalanced, with a large number of negative cases compared to positive cases.  The cross-validation results show that the best hyperparameters for the model are neighbors= 3, 5 and weights = uniform, distance and p = 1, 2. These hyperparameters were chosen because they resulted in the best cross-validation score during the hyperparameter tuning process.  The main underperforming cases/observations are the false negatives and false positives. These misclassifications may be caused by the imbalanced nature of the dataset, as there are many more negative cases than positive cases. This could lead the model to have a bias towards predicting the majority class (i.e., negative) and thus perform poorly on the positive class. Additionally, the features used to train the model may not be sufficient to capture all the important information related to predicting the target variable. In future experiments, it may be useful to explore other models, such as ensemble methods or deep learning models, and to investigate additional features that may improve the performance of the model.  Based on the results achieved and the overall objective of the project, there are several potential next steps and experiments that can be pursued:   1. Collect more data: The current dataset contains a relatively small number of positive samples, which may have contributed to the relatively low recall of the model. Collecting more data, especially positive samples, could help to improve the performance of the model. 2. Explore alternative models: While the Gradient Boosting model performed well, it may be worth exploring other models such as Neural Networks or Random Forests to see if they can achieve even better performance. 3. Hyperparameter Tuning: Further hyperparameter tuning of the Gradient Boosting model could also potentially improve its performance. 4. Feature engineering: While the current set of features performed well, additional feature engineering could help improve the accuracy of the model. For example, adding interaction features or feature selection based on correlation analysis could be explored. 5. Ensemble models: Ensemble models can be used to combine the outputs of multiple models to obtain a better overall performance. This can be done by either combining models with different architectures, or by combining models that have been trained on different feature subsets. 6. Deployment: If the model achieves the required outcome for the business, it can be deployed into production. This will involve integrating the model into the existing software infrastructure, and possibly developing a user interface to allow users to interact with the model. The deployment process should include testing and validation to ensure that the model is performing as expected in the production environment.   Overall, pursuing the above steps could potentially improve the performance of the model and provide more accurate predictions for the business. The priority and expected gains of each step will depend on the specific business objective and requirements. |