Building and Tuning Classification Models for Happiness Prediction

Introduction

This technical report outlines the process of building and tuning classification models for predicting happiness levels using the 'ACME-HappinessSurvey2020.csv' dataset. We employ various classification algorithms and Python's Scikit-Learn library to accomplish this task. The primary objective is to identify the best-performing model for predicting happiness levels accurately.

Libraries Used

The following libraries were employed for this project:

- Pandas: Utilized for data loading and manipulation.
- Scikit-Learn: Employed for machine learning tasks including model selection, data preprocessing, and evaluation.
- XGBoost: An optimized gradient boosting library.
- Warning: Used to suppress warning messages.

Data Loading and Preprocessing

The initial steps involve loading the dataset and preparing it for machine learning.

1. Data Loading: The dataset is loaded into a Pandas DataFrame named `df`. It contains the survey data, with columns representing features and a target variable indicating happiness levels.

2. Data Preprocessing: We scale the features using the StandardScaler to ensure that all features have a similar scale. Scaling helps prevent features with large values from dominating the learning process.

Train-Test Split

To evaluate the model's performance, we split the data into training and testing sets. An 80/20 split is employed, with 80% of the data allocated for training and 20% for testing.

Model Pipeline

We construct pipelines for various classification algorithms to streamline the preprocessing steps and the classifier into a single object. The pipelines are created for the following classifiers:

- Random Forest
- Gradient Boosting
- XGBoost
- Support Vector Classifier (SVC)
- Logistic Regression
- K-Nearest Neighbors (KNN)

Hyperparameter Grid

For each classifier, we define a grid of hyperparameters to be tuned during the model selection process. These hyperparameters vary depending on the specific classifier and include parameters like the number of estimators, learning rate, kernel type, and more.

Model Training and Tuning

We iterate through each classifier, apply 10-fold cross-validation, and tune the hyperparameters using GridSearchCV. This process involves finding the best combination of hyperparameters for each classifier.

Model Evaluation and Selection

We evaluate the performance of each model on the test data and select the model with the highest accuracy. The accuracy metric is used to determine the model's predictive performance on unseen data.

Conclusion

This technical report has outlined the steps involved in building and fine-tuning classification models for predicting happiness levels based on the 'ACME-HappinessSurvey2020.csv' dataset. The best-performing model, as determined by its accuracy on the test set, is selected for predicting happiness levels.

It is important to note that the choice of the best model depends on its accuracy, and further enhancements can be explored by experimenting with different algorithms, conducting feature engineering, or collecting additional data. The selected model can serve as a tool for predicting happiness levels based on input features and has the potential to provide valuable insights for various applications.

Improving Classification Model for Happiness Prediction

Introduction

This technical report focuses on enhancing the accuracy of a classification model for predicting happiness levels using the 'ACME-HappinessSurvey2020.csv' dataset. We aim to mitigate potential overfitting issues and refine the model's hyperparameters for improved performance.

Libraries Used

We utilized several Python libraries for this task, including:

- Pandas: Data loading and manipulation.
- Scikit-Learn: Model selection, data preprocessing, and evaluation.
- XGBoost: An optimized gradient boosting library.
- Matplotlib: Visualization of results.

Data Loading and Preprocessing

We initiated the process by loading the dataset and preparing it for machine learning:

1. Data Loading: The dataset, stored in 'ACME-HappinessSurvey2020.csv', was loaded into a Pandas DataFrame named `df`. It contains survey data, with columns representing features and a target variable indicating happiness levels.

2. Data Preprocessing: The features were standardized using the StandardScaler to ensure uniform scale. Standardization helps prevent features with large values from dominating the learning process.

Model Refinement

To enhance the model's accuracy and address potential overfitting, we adopted several strategies:

Stratified K-Fold Cross-Validation: We used a StratifiedKFold with 5 splits to maintain the class distribution during cross-validation. This ensures robust evaluation.

XGBoost Classifie: We employed the XGBoost classifier, setting `use_label_encoder` to False and `eval_metric` to 'logloss'.

Hyperparameter Tuning: A refined parameter grid was defined to search for optimal hyperparameters. The grid included parameters such as `n_estimators`, `learning_rate`,

`max_depth`, `colsample_bytree`, `subsample`, `gamma`, `reg_lambda`, and `reg_alpha`. Increased regularization was introduced to mitigate overfitting.

Grid Search

A GridSearchCV was performed with the defined parameter grid, aiming to discover the best combination of hyperparameters. The evaluation metric used was accuracy, and the search was conducted in parallel (`n_jobs=-1`) for efficiency.

Model Training and Testing

Following hyperparameter tuning, the dataset was re-split into training and testing sets (80/20 split) to train the final model. The best model, determined by the grid search, was trained on the training data.

Model Evaluation

The final model was evaluated using the test set, and its performance was assessed using accuracy. Additionally, a classification report and a confusion matrix were generated to provide detailed insights into the model's performance.

Conclusion

This technical report has outlined the process of improving a classification model for happiness prediction. Through hyperparameter tuning, regularization, and thorough evaluation, we aimed to enhance accuracy and mitigate overfitting. The selected model demonstrates its potential to predict happiness levels effectively based on input features, providing valuable insights for various applications.

Building and Evaluating Multiple Classification Models for Happiness Prediction using improved parameters from the above model

Introduction

This technical report focuses on building and evaluating multiple classification models to predict happiness levels using the 'ACME-HappinessSurvey2020.csv' dataset. The goal is to identify the most accurate model and compare its performance against other models. We also used the improved hyperparameters obtained from a previous code iteration to enhance model performance.

Libraries Used

We utilized several Python libraries for this task, including:

Pandas: Data loading and manipulation.
Scikit-Learn: Model selection, data preprocessing, and evaluation.
XGBoost: An optimized gradient boosting library.
Matplotlib: Visualization of results.

Data Loading and Preprocessing

The process began with loading the dataset and preparing it for machine learning:

1. Data Loading: The dataset, stored in 'ACME-HappinessSurvey2020.csv', was loaded into a Pandas DataFrame named `df`. It contains survey data, with columns representing features and a target variable indicating happiness levels.

2. Data Preprocessing: The features were standardized using the StandardScaler to ensure uniform scale. Standardization helps prevent features with large values from dominating the learning process.

Model Selection and Hyperparameter Tuning

To identify the most accurate model and refine its hyperparameters, we followed these steps:

Stratified K-Fold Cross-Validation**: We employed a StratifiedKFold with 5 splits to maintain the class distribution during cross-validation. This ensures robust evaluation.

Model Initialization**: We initialized six different classification models, including XGBoost, Random Forest, Gradient Boosting, Support Vector Classifier (SVC), Logistic Regression, and K-Nearest Neighbors (KNN).

Hyperparameter Grids**: For each model, we defined specific hyperparameter grids to search for optimal values. These grids included parameters such as `n_estimators`, `learning_rate`, `max_depth`, `colsample_bytree`, `subsample`, `gamma`, `reg_lambda`, and `reg_alpha`. Increased regularization was introduced to mitigate overfitting.

Model Evaluation

We evaluated each model using the following steps:

Grid Search: We employed GridSearchCV to perform hyperparameter tuning for each model, aiming to find the best combination of hyperparameters that maximizes accuracy. The evaluation metric used was accuracy.

Model Training and Testing: After hyperparameter tuning, the dataset was re-split into training and testing sets (80/20 split) to train the final model. The best model, determined by the grid search, was trained on the training data.

Test Accuracy: We calculated the accuracy of each model on the test set to assess its predictive performance.

Results and Visualization

The results were visualized using a bar chart that displays both validation and test accuracies for each model. The chart also includes a red dashed line indicating the baseline accuracy of 0.73. The baseline represents the accuracy achieved by a simple model that predicts the majority class, providing a reference point for model performance.

Conclusion

This technical report has demonstrated the process of building and evaluating multiple classification models for predicting happiness levels. By refining hyperparameters and employing cross-validation, we aimed to improve accuracy and mitigate overfitting. The selected

model, with its hyperparameters fine-tuned, demonstrates its potential to predict happiness levels effectively based on input features, providing valuable insights for various applications.

The visualization comparing the validation and test accuracies highlights the model's performance relative to the baseline accuracy, aiding in the selection of the most suitable model for the happiness prediction task.