BDM 3014 - Introduction to Artificial Intelligence 01

Project Report

PROJECT OVERVIEW

The KDD Cup 1998 dataset is a well-known dataset used for the Second International Knowledge Discovery and Data Mining Tools Competition. This competition was held in conjunction with KDD-98, the Fourth International Conference on Knowledge Discovery and Data Mining. The dataset was used for a regression problem related to direct mailing, with the objective of estimating the return in order to maximize donation profits.

In this competition, participants were tasked with developing predictive models to estimate the donation response from individuals who were targeted with direct mail solicitations. The dataset contained various features related to individuals' demographics, past donation behavior, and other relevant information. Participants were supposed to build regression models that could accurately predict the amount of donation or response probability for each individual.

The goal of the competition was to encourage researchers and practitioners to develop effective data mining and machine learning techniques for solving real-world problems related to direct marketing and donation campaigns.

PROBLEM

The dataset encompasses details concerning donors who have previously responded to the organization's direct mailing initiatives. The task involves forecasting both the likelihood of future donations from these donors and the potential donation amounts. This prediction relies on a combination of historical behaviour and demographic attributes. The objective is to enhance the organization's profit by minimizing mailing costs while concurrently maximizing revenue generated from donations.

DATASET COLLECTION

The KDD Cup 1998 data set is a collection of data from a non-profit organization that helps US veterans. The competition organizers provided a dataset that comprised a substantial 95412 data points, each associated with an extensive array of 481 features. The dataset was meticulously crafted to mirror the complexities inherent in real-world scenarios, ensuring the competition's relevance to practical applications. The dataset have numerical and categorical values which contains details of donors .

http://archive.ics.uci.edu/dataset/129/kdd+cup+1998+data

DATA PREPROCESSING:

1. Unique Values and Data Information:

   - The code starts by extracting the unique values from the 'INCOME' column of the featured data and printing them.

   - It then prints information about the featured data using the `info()` method.

2. Binning and Labeling:

   - Binning boundaries (bins) are defined, and corresponding labels are assigned for categorical conversion of 'INCOME'.

   - The code applies binning to the 'INCOME' column using `pd.cut` and creates a new 'INCOME_category' column with categorical labels.

3. Label Encoding:

   - An instance of `LabelEncoder` is initialized.

   - Label encoding is applied to the 'INCOME_category' column, creating an 'INCOME_encoded' column with encoded numeric labels.

4. Data Copy and Column Deletion:

   - A copy of the featured data is created as `df_copy`.

   - Columns specified in the `columns_to_delete` list are removed from the copy.

5. Standard Scaling:

   - The features in `df_copy` are standardized using `StandardScaler`.

6. PCA Transformation:

   - Principal Component Analysis (PCA) is applied to the standardized data, reducing it to six principal components.

7. Data Splitting:

- The target variable `y` is defined as the 'INCOME_encoded' column.
- `X` is set as the PCA-transformed features.

8. Train-Test Split:

- The PCA-transformed features and target variables are split into training and testing sets.

MODEL SELECTION:

1. Classifier Import and Initialization:

- The code imports classifiers: `RandomForestClassifier`, `SVC` (Support Vector Classifier), and `GradientBoostingClassifier`.
- For each classifier, hyperparameter grids for tuning are defined.

2. Model Training and Prediction:

- Each classifier is trained and evaluated individually using the training data (`Xpca_train` and `ypca_train`).
- Predictions are made on the testing data (`Xpca_test`) using each trained classifier.

3. Metrics Calculation and Display:

- Custom function `calculate_metrics` calculates metrics (accuracy, precision, recall, F1 score) for each classifier's predictions.
- Metrics for each classifier are displayed, providing insights into the performance of individual models.

4. Ensemble Creation and Evaluation:

- A hard-voting ensemble of the classifiers is created using `VotingClassifier`.
- The ensemble is trained and evaluated on the testing data, and metrics are calculated for the ensemble's predictions.

<u>MODEL EVALUATION</u>:

1. LIME (Local Interpretable Model-agnostic Explanations):

   - The code continues by introducing LIME.

   - Categorical labels are converted to numeric using `LabelEncoder` for compatibility with LIME.

   - `LimeTabularExplainer` is instantiated to explain the ensemble's predictions using the trained ensemble's `predict_proba`.

2. LIME Explanation Visualization:

   - A specific prediction (index 0) is chosen to be explained using LIME.

   - LIME explanation is generated using `explainer.explain_instance`.

   - The explanation is visualized and displayed in the notebook.

3. Hyperparameter Tuning and Cross-Validation:

   - For each classifier, hyperparameter tuning is performed using `GridSearchCV` and cross-validation.

   - Best parameters and cross-validated accuracy are displayed.

   - The tuned classifier is then fitted to the training data, predictions are made, and metrics are calculated.

4. Ensemble Cross-Validation and Evaluation:

   - Cross-validation is performed on the ensemble using `cross_val_score` to assess its performance.

   - The ensemble is trained and evaluated on the testing data, and metrics are calculated.

MECE Table

Model	Issues	Solutions
Random Forest Classifier	– May overfit if hyperparameters are not tuned – GridSearchCV can be time-consuming	– Perform hyperparameter tuning using GridSearchCV to find optimal parameters – Monitor training accuracy and validation accuracy to detect overfitting
Support Vector Machine (SVM) Classifier	– Might not perform well with linear kernel if data is not linearly separable- Could be computationally expensive for large datasets	– Consider using non-linear kernels (e.g., 'rbf') to handle non-linearly separable data – Experiment with different values of the regularization parameter 'C'
Gradient Boosting Classifier	– Consider using non-linear kernels (e.g., 'rbf') to handle non-linearly separable data	– Perform hyperparameter tuning using GridSearchCV to find suitable values for max_depth and n_estimators – Consider using robust loss functions or outlier handling techniques if dealing with outliers
Voting Classifier Ensemble	– May not work well if individual models have very different performance – Some models might dominate the ensemble's decision	– Evaluate individual models' performance before combining them – Experiment with different voting strategies ('hard' or 'soft') to combine predictions

Model	Issues	Solutions
		– Assign appropriate weights to models based on their performance

Project Work Table

Task	Model	Status	Notes
Stacked Ensemble	support vector	Done	Accuracy: 0.662227602905569 Precision: 0.63939209765838 Recall: 0.662227602905569 F1 Score: 0.6453252953702557
	gradient boosting	Done	Accuracy: 0.8018563357546409 Precision: 0.8037579511703015 Recall: 0.8018563357546409 F1 Score: 0.798342638423913
	Random Forest	Done	Accuracy: 0.8668280871670703 Precision: 0.8672405183700865 Recall: 0.8668280871670703 F1 Score: 0.8643421152068136
	Ensemble	Net Perf Calculated	Accuracy: 0.8071025020177562 Precision: 0.8135810447462313 Recall: 0.8071025020177562 F1 Score: 0.8018647620455103
Interpretation		Done	Model results and insights interpreted and documented.
Model Tuning	Issue: Might not perform well with linear kernel if data is not linearly separable- Could be computationally expensive for large dataset	Solved	Solution: Consider using non-linear kernels (e.g., 'rbf') to handle non-linearly separable data
	Issue: Consider using non-linear kernels (e.g., 'rbf') to handle non-linearly separable data	Solved	Solution: Perform hyperparameter tuning using GridSearchCV to find suitable values for max_depth and n_estimators
	Issue:If there are too many trees or if the trees are too deep, Random Forest models are prone to overfitting..	Solved	Solution: Hyperparameter tweaking has been used to restrict tree depth and manage the forest's tree population.
Deployment and Demo	Deployment Done	Done	Streamlit

Task	Model	Status	Notes
	Demo Done	Done	We recorded the demo on meet, of the application deployed on streamlit.
GitHub Repository Link	Each member performs a code check-in once each week.	Done	All team members checked the code
	A different individual inspected all code deployment from development to testing and provided comments, etc.	Done	Code was reviewed by all team members

Link to the Trello Board:
https://trello.com/invite/b/WBr926Jt/ATTIc2c89d822daacf013117d4f8df3dc6df8A6B696B/aiproject

Link to Github:
https://github.com/ArchanaVInfy/AI_Project

Group members
ALEENA BINOY
ARCHANA VIJAYAN
ANOOP JAYA PRAKASH
MANIKANTAN S
SHARAN SARA SHAJI