Exploring Supervised Learning Techniques:

A Comparative Study of Decision Trees, Naive Bayes, and Ensemble Models

Introduction:

This report discusses the performance of two classifiers: Decision Tree and Naive Bayes and two ensemble classifiers Bagging with decision trees (or a random forest) and Boosting with decision stumps.

Introducing to the Data Set:

The "Bank Marketing Data Set" from the UCI Machine Learning Repository is related to a Portuguese banking institution's direct marketing campaigns (phone calls).
The classification goal is to predict if the client will subscribe to a term deposit (variable y).

Data pre-processing

- No null values were found
- Some categorical columns like ‘default’, ‘marital’, ‘housing’, ‘loan’, etc contained a lot of ‘unknown’ values which accounted for nearly 25% of the total number of recorded instances. Thus, these could neither be removed (as it would lead to a significant loss of rows of data) nor replaced by another value (as they are categorical and each row represents a different case, making generalization a little difficult).
- One-hot encoding was applied to all the categorical columns.

Decision Tree Classifier:

- After one-hot encoding of the data, we built and trained a Decision Tree without any restrictions.
- We got an accuracy of 88.55 % and decided to tackle a common Decision Tree problem - overfitting, by pruning.
- We fix the value of ccp alpha ccp_alpha = 0.0002.

Naive Bayes Classifier:

Gaussian Naive Bayes:

- We used Gaussian Naive Bayes Classifier in particular because the features dataset consisted of a mix of categorical (in 0s and 1s after one hot encoding) and numerical (real numbers) attributes.
- An accuracy of 86.69 % was achieved without any feature selection. Hence, we decided to proceed with feature selection in order to improve the performance metrics.

Feature Selection:

- Since some of the categorical columns had 'unknown' as values, we decided to use feature selection and compare their importance.
- For categorical columns, we used the Chi-square test to find out how much each feature affected the target variable. Similarly, for the numerical columns, the F-test (used in ANOVA) was performed.
- Accordingly, some features which scored really low on the feature importance graph were removed from the feature space. And Gaussian Naive Bayes was performed on the rest.
- However, the accuracy score achieved was 84.85 % and so, we decided to perform hyperparameter optimization to see if the performance of the classifier could be improved.

Hyperparameter Tuning:

- GridSearchCV was used to perform cross-validation on each selected set of hyperparameters to obtain the optimized parameters. After that, the accuracy improved to 90.62 %.
- Number of cross-validations tried for each selected hyperparameter = 10.

Random Forest Classifier:

- After one-hot encoding of the data, we built and trained a Random Forest Classifier without any restriction.
- We got an accuracy of 91.6 %.

Hyperparameter Optimisation:

- We know that by setting the values of `max_depth`, `min_samples_split`, and `min_sample_leaf` we can restrict our decision tree from overfitting.

- But we need to know the optimal values of the parameters in order to get the maximum accuracy score as well as not overfit the training data.

Finding the optimal value:

- We used the GridSearchCV function to perform a 5-fold cross-validation on the training data to find the optimum values for the parameter.
- Then we built and fit a Random Forest based on the optimum parameter found using GridSearchCV and got an accuracy score of 90.35 %.

Boosting with decision stumps:

- After one-hot encoding of the data, we built and trained an Ada Boost Classifier with base_estimator as Decision Tree Classifier with max_depth = 1 without any other restriction.
- We got an accuracy of 91.22 %.

Hyperparameter Optimisation:

- We know that by setting the values of `n_estimator`, `learning_rate`, and `algorithm` we can restrict our decision tree from overfitting.
- But we need to know the optimal values of the parameters in order to get the maximum accuracy score as well as not overfit the training data.

Finding the optimal value:

- We used the GridSearchCV function to perform a 5-fold cross-validation on the training data to find the optimum values for the parameter.
- Then we built and fit a Ada Boost Classifier with base_estimator as Decision Tree Classifier with max_depth = 1 based on the optimum parameter found using GridSearchCV and got an accuracy score of 91.11 %.

Comparative Evaluation

The final performance metrics of both the classifiers in both the tasks are given below:

Classifier	Accuracy	Recall	Precision	Memory (MiB)	Running Time (Sec)
Decision Tree	0.90	0.62	0.80	220.76	3.12
Naive Bayes	0.90	0.59	0.75	258.12	1.61
Random Forest Classifier	0.89	0.62	0.73	362.40	10.44
Random Forest Classifier with Grid Search CV	0.90	0.58	0.83	333.42	501.06
Boosting with decision stumps	0.90	0.58	0.83	333.84	415.34
Boosting with decision stumps with Grid Search CV	0.90	0.59	0.82	245.95	920.05