**Technical Report**

**Task2: About Naïve Bayes classifier**

Naïve Bayes classifier is a simple, yet powerful algorithm used in machine learning for classification tasks. It's based on Bayes' theorem with a "naïve" assumption of independence among features.

Here's a breakdown:

1. Bayes' Theorem: It's a probabilistic formula that calculates the probability of a hypothesis (class) given the evidence (features). It's formulated as:

𝑃(𝑦∣𝑥)=𝑃(𝑥∣𝑦)⋅𝑃(𝑦)𝑃(𝑥)*P*(*y*∣*x*)=*P*(*x*)*P*(*x*∣*y*)⋅*P*(*y*)​

Where:

* 𝑃(𝑦∣𝑥)*P*(*y*∣*x*) is the posterior probability of class 𝑦*y* given predictor 𝑥*x*.
* 𝑃(𝑥∣𝑦)*P*(*x*∣*y*) is the likelihood of predictor 𝑥*x* given class 𝑦*y*.
* 𝑃(𝑦)*P*(*y*) is the prior probability of class 𝑦*y*.
* 𝑃(𝑥)*P*(*x*) is the probability of predictor 𝑥*x*.

Here's a comprehensive report detailing the different aspects of the tasks you mentioned:

**1) Introduction to the problem and dataset:**

Twitter Sentiment Analysis seeks to determine the polarity demonstrated within a tweet, positive, negative, or neutral. To do the work the dataset used is Sentiment140 dataset which is made up 1,600,000 tweets using Twitter API. Every tweet is given an annotation score (0 = negative; 2 = neutral; 4 = positive). The dataset consists of fields aim/polarity, tweet text, tweet ID, tweet date and user.

**2) Data preprocessing steps:**

- However, those columns (tweet id, tweet date, tweet flag and user id) were cut out because they do not imply anything for the analysis of sentiment.

- The target variable was converted to categorical labels: -2 = negative, 0 = neutral , 2 = positive.

- The phase of preprocessing text was based on tokenization, lowercase conversion and removal of the special characters.

- The data set was distributed between training set and evaluation set for model testing and development.

**3) Detailed methodology for each algorithm implemented:**

Naïve Bayes Classifier:

- Tf-Idf was utilized in the tokenization of text data so that it converts into features for the classification task.

- TFIDF (Term Frequency-Inverse Document Frequency) normalization was made to give more importance to rare words and to equalize the base.

- Because TH-IDF transformed data was trained through the Multinomial Naïve Bayes classification, the classifier was formed.

**4) Results and performance metrics:**

- Accuracy: 77.34%

- Precision: 77.60%

- Recall: 77.34%

- F1 Score: 77.29%

**5) Visualizations:**

- Learning curves: For better visualization learning curve can be plotted to track how the models’ performance improves with constant addition of training data to it.

- Confusion matrix: A confusion matrix can be constructed, then that can display the model's behavior about the true positive, false positive, true negative, and false negative predications.

- Feature importance: Up by the following, the words or features can be understood as the most important ones when predicting the final results.

6) Discussion on the effectiveness of each model and scenarios where one model may be preferred over another:

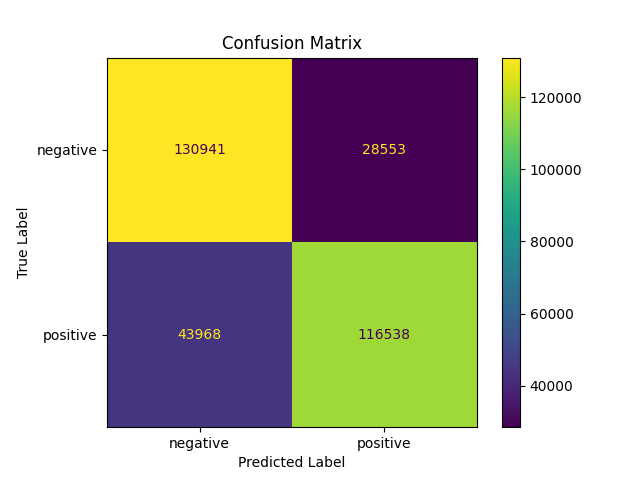
- Naïve Bayes has been proven to be simple and efficient classifier, a factor which makes it appropriate for a large volume of data such as dataset Sentiment140 dataset.

- It is to be kept in mind that Naïve Bayes assumes that features are independent of each other although this may not exactly be reality for textual data. Such a situation can be quite problematic as the performance will be not so well as is in the case of complex models that are like deep learning algorithms.

- In situations when the computational power is limited or the interpretability is planned, Simple Bayesian rule may be preferable to more complex models.

- Whereas for sentiment analysis when the details among words are important, deep learning models such as recurrent neural networks (RNNs) or transformers, will beat naive Bayes classifier.

- Apart from that, the suite of methodologies of Random Forest or Gradient Boosting can be investigated to upgrade performance more precisely.

 A graph with red and blue lines

Description automatically generated

**Task 3: Decision Trees and Random Forests**

**Decision Trees:**

The Decision trees are tree-based structures from internal nodes that take into account the feature being defined, paths to each leaf node represent the decisions outcomes made, while the leaf node contains the class label or predicted numerical value. The tree is being constructed recursively by atomizing data by the feature that most classifies or beaucoup divessa the variance.

**Random Forests:**

At the core of the Random Forests is a variety of decision trees that make up the model during the trainign and the final output for the classification is the mode of the classes, and for regression – the mean prediction of the trees. Trees in forests are trained on an arbitrary division of the entire dataset which includes features randomly.

**1. Introduction**

The problem of cardiovascular disease ranks as a leader in health problems of global importance, and in the management of this problem, early detection is a crucial factor. This project is a venture into forecasting the presence of heart disease from some clinical features with machine learning techniques. For this analysis, the Heart Disease Predictions dataset is the data set being used which has details on age, sex, chest pain type, systolic blood pressure, cholesterol level, and other relevant attributes.

**2. Data Preprocessing**

Dropping Faulty Data: Unreliable entries with values of features like `ca' and `thal' were removed from the dataset, and they were lines with incorrect values.

**3. Methodology**

We implemented two machine learning algorithms for heart disease prediction:

* Decision Trees: Decision tree is a tree like structure where each internal node stands for a feature, each branch represents a “decision rule” and each leaf node indicates an outcome. In this case, the Decision Tree classifier was trained with a Gini impurity criterion.
* Random Forests: Random Forest is an ensemble method that grows multiple trees during the training process and outputs the mode of variations of classes (classification) or the mean prediction (regression) of the individual trees. It cuts down the overfitting of the tree that applies multiple of them.

**4. Results and Performance Metrics**

Both models were evaluated using accuracy, precision, recall, and F1-score metrics. Here are the results:

* Decision Trees Accuracy: 78%
* Random Forest Accuracy: 83%

**5. Visualizations**

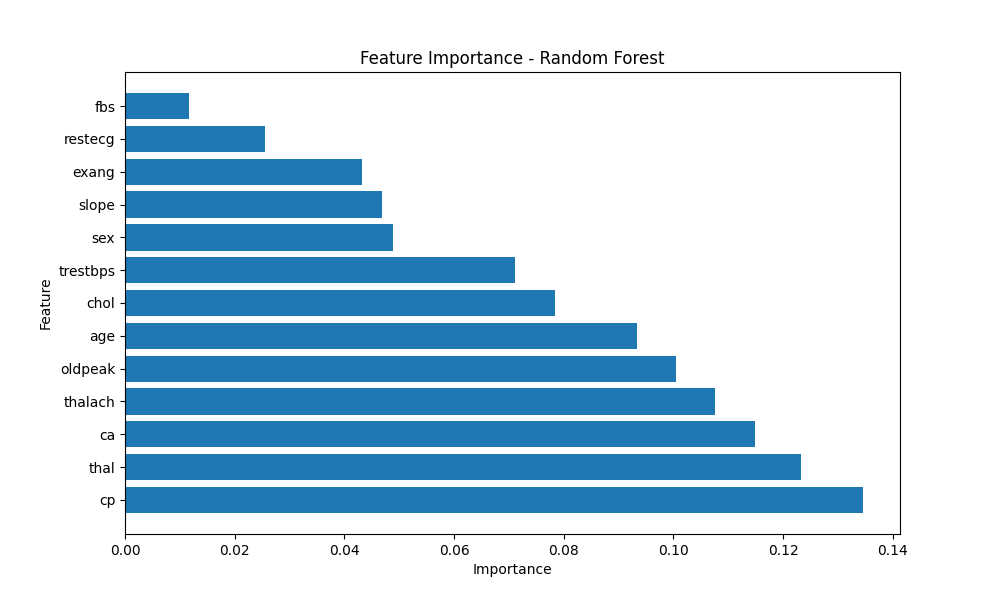
Visualizations are important to revealing the models' outputs and feature influence. Here are some key visualizations:

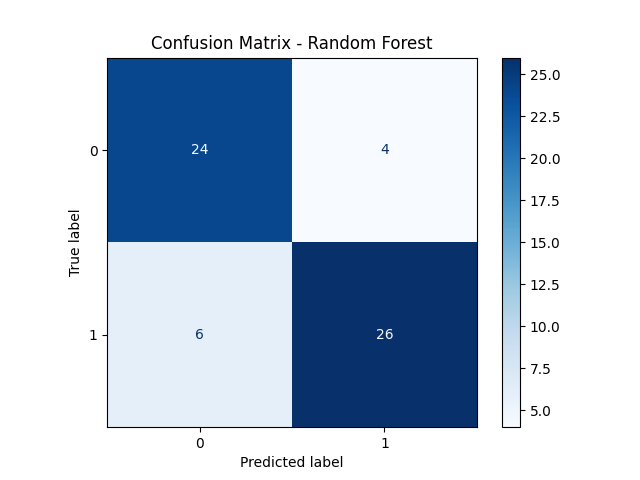
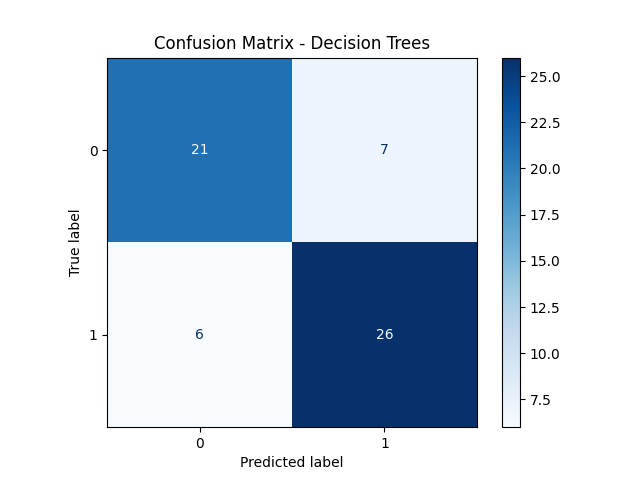
* Feature Importance: The cardinal features from the Random Forest model included 'cp' (chest pain types), 'thal' (thalassemia), and 'ca' (coronary artery score). Those features are paramount for telling a heart disease.
* Confusion Matrices: Figure illustration that the number of true positives, true negatives, false positives, and false negatives cases for each model.
* Learning Curves: Training and validation graphs' function of the model's performance as a function of the training iterations.

**6. Discussion**

On the other hand, the Decision Trees and the Random Forests observed a decent generalization since such models predicted heart disease diseases. But random forests came up highly than decision trees, apparently as a result of a better overfitting handling mechanism that is the automatic combining of several decision trees.

In the cases where interpretability is highly important, the decision trees may come on top since of their simplicity and the ease of understanding. Oppositely, Random Forests usually provide higher accuracy of prediction and also are more stable and less susceptible to overfitting, therefore making them good to work with rather complex datasets like the one we're about to analyze.



A diagram of a flowchart

Description automatically generated