Machine Learning Competitive Project
Kaggle Competition link: https://www.kaggle.com/competitions/income2022f
Kaggle Username & link: dhruvilshah2 & https://www.kaggle.com/dhruvilshah2
Name: Dhruvil Shah
uID: u1420007

Problem Statement: We have been provided with two datasets, train_final.csv and test_final.csv. The aim of this competition is to perform binary classification using various Machine Learning algorithms to predicted whether the income of an individual would be higher than $50K based on various factors defined below:

1. age: continuous
2. workclass: {Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked}
3. fnlwgt: continuous
4. education: {Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool}
5. education-num: continuous.
   marital-status: {Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse}
6. occupation: {Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces}
7. relationship: {Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried}
8. race: {White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black}
9. sex: {Female, Male}
10. capital-gain: continuous.
11. capital-loss: continuous.
12. hours-per-week: continuous.
13. native-country: {United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands}

To do so, I considered multiple approaches, namely Logistic Regression, Naive Bayes, Random Forest, K-nearest neighbours, Decision Tree, Support Vector Machines. After careful consideration of each method based on the following steps I decided on implementing Random Forest, Decision tree and Extreme Gradient Boosting:

1. Understanding project goal: We require a binary prediction of the income based on their information, this means that it is a supervised classification problem.
2. Analyze Your Data Based on Size, Processing, and Required Annotation
3. Analyze the Training Time and Speed.
4. Analyze Your Data's Linearity

5. Decide on the Features and Parameters to Include.

Now let's talk about the steps that I followed to train the models:

1. Data Preprocessing: Our data consists of multiple missing values and requires the conversion of categorical data to continuous data.

   Replacing missing values: Our training data has missing values represented by "?" in the following quantities: workclass: 1437(5.75%), occupation 1442(5.77%), native.country 427(1.71%) and our test data has:workclass 1362(5.71%), occupation 1367(5.73%), native.country 430(1.80%). It is impertinent to replace these unknown values. I replaced them with np.NaN.

   Data Conversion(one-hot vectors): Columns workclass, education, marital-status, occupation, relationship, race, native-country have categorical data value and we need to convert it into continuous datatype to make predictions

2. Feature Selection: Now we need to select the best features which we can use to make predictions, to do this we need to calculate correlation and sort the features according to their absolute values.
3. Now we need to check the minimum number of features we need to make an accurate prediction, we do this by finding the cross validation score. This shows us that we need atleast 4 features.
4. Now we move to the most important step i.e prediction
   a. **Random Forest:** Random forest is an ensemble of decision trees. This is to say that many trees, constructed in a certain "random" way form a Random Forest. Each tree is created from a different sample of rows and at each node, a different sample of features is selected for splitting. Each of the trees makes its own individual prediction. These predictions are then averaged to produce a single result.

      We use 1000 trees in the forest, while taking the gini index as the criterion. The minimum number of samples required for a split an internal node are set to 320. This trains our model to which we fit to the train dataset. We then pass our test dataset through the trained model to get our prediction. To submit the prediction, I converted the output into a .csv file with headers {ID, Prediction}.

   b. **Decision Tree:** A decision tree is a supervised machine learning tool that may be used to classify or forecast data based on how queries from the past have been answered. The model is supervised learning in nature, which means that it is trained and tested using data sets that contain the required categorisation.

      The decision tree might not always offer a simple solution or choice. Instead, it might give the data scientist choices so they can choose wisely on their own. Decision trees mimic human thought processes, making it generally simple for data scientists to comprehend and evaluate the findings.

Each entry in the dictionary that contained the data included a list of associated entries that had the values for each row. The position function, which returns the index of the specified attribute name, can therefore be used to quickly determine the position of each attribute. The data processing for the decision tree implementation was done in this.

First, a tree has to be formed from the root up in order to forecast a class label. The main method used to forecast the result involved comparing the values of the root property with the target's attribute, then following the branch corresponding to that value and jumping to the next node. A measure was required in a decision tree to determine which attribute to split on the tree. Information gain, Gini index, and majority error were used for this. For each attribute, entropy—a measure of the dataset's degree of uncertainty—is computed. The attribute with the lowest is selected for splitting.

c. **Neural Networks**
d. **Extreme Gradient Boosting:**

A machine learning method called gradient boosting is used, among other things, for classification and regression tasks. It provides a prediction model in the form of an ensemble of decision trees-like weak prediction models. The resulting technique, known as gradient-boosted trees, typically beats random forest when a decision tree is the weak learner. The construction of a gradient-boosted trees model follows the same stage-wise process as previous boosting techniques, but it generalizes other techniques by enabling the optimization of any differentiable loss function.

An ensemble method called XGBoost uses the gradient boosting algorithm. The performance of the models and computation speed are the key focuses of XGBoost. When compared to other gradient-boosting technique implementations, XGBoost is incredibly quick. The method is frequently used in Kaggle contests.

- n estimators, which defines the number of boosting rounds, is one of the two most crucial parameters for XGBoost algorithms. I tried using boosting rounds of 100, 130, and 150.
- max_depth – maximum tree depth for weak learners. I tried varying the max_depth attribute between 5 to 10.

Future Plan

If I had more time I would try and implement various types of algorithms and compare their accuracies. I would also try to reduce dimensionality of our data.