Prediction of Adult Census Income dataset using different classification models

Abstract

In this project, We have performed a detailed comparison of 12 different classification models for the Adult Census Income dataset to predict whether a person's income is over 50000 USD or not. We have used 12 different classification models and performed different preprocessing techniques on the dataset and hyperparameter tuning for the models to further improve the accuracy in the prediction of labels mentioned.
We have observed 86% overall accuracy for default values and improvement of over 2% (88.3% overall accuracy) after the optimization of the models. The results of the experiments show that a similar set of preprocessing and classification models can be used for comparing many prediction problems similar to the one presented in the project.

Introduction

The dataset we have used here is Adult Census Income which is used for prediction of income, whether the income of a person is greater than 50000 USD per annum or not.

The dataset is taken from the **1994 Census bureau database.**
The data is extracted using 2 conditions :
1. Age of person is greater than 16 and less than 100
2. The final weight of a person is greater than 1

The **objective** of the project is to correctly classify whether a person makes over 50000 USD a year using various classification models and observe their accuracy in prediction.

We have 2 classification labels: <=50K and >50K.

There are 15 Columns (Features) taken into account :
age - Age of person
workclass - Class of work (Private, Self-employed, Government employee, etc.)
fnlwgt - Final Weight (Described below)
education - Education level of the person (10th, grad, postgrad, etc.)
education.num - Education level in numeric format (Ordinal)
marital.status - Marital status of the person (Single, Married, Divorced, etc.)

occupation - Type of work (Sales, clerk, Executive Manager, etc.)
relationship - Relationship of the person
race - Race (White, Black, Brown, etc.)
sex - Male or Female
capital.gain - Capital gain of person
capital.loss - Capital loss of person
hours.per.week - How many hours does the person work a week
native.country - Native country
income - Class label (<=50K and >50K)

As this is a Classification task, There are several Machine learning classification models that can be used for prediction, especially binary class prediction models. So we have used the 12 most common classification models generally used for binary classification and compared their accuracy.

The **motivation** of the project is to check whether a person's attributes such as nationality, gender, race, occupation matter in terms of the income they get and how does each of the attributes factor in the income, which set of features contributes the most for earning more salary, etc.

Relevant papers
Ron Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid", Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996. (PDF)

Research

The topic for research is **Feature Selection.**
There are various techniques for feature selection, I have implemented 3 of them in my code.

1. Reducing features with low variance
This is a feature reduction technique in which the model removes the features whose variance doesn't meet a particular threshold.
e.g.: If some feature has the same value in all the data points, Then that feature is not contributing to the classification model and hence can be removed.

As an example, In our dataset, most of the rows have value 'USA' in 'native country' column, We can set a threshold for the same and remove the feature if 90% of rows have the same value.

2. Univariate feature selection
Univariate feature selection works on the selection of best features based on tests.

In UFS, There are multiple techniques for selecting the best features,
1. **Select K best** - It simply keeps the top-scoring features and eliminates the rest.
2. **SelectPercentile** - Similar to K best, but instead of specifying the count, We can specify the percentage of the top-scoring features we want to keep.
3. **Generic Univariate Select** - You can configure the strategy for Univariate selection (Hyperparameter tuning)

3. Tree-Based feature selection
In Tree-based feature selection, We can specify the algorithm (Extra trees in our case) and it builds the trees and calculates the importance of features based on their reduction in uncertainty. So basically, the larger the reduction in uncertainty the more important is the feature.
The algorithm we have selected will decide the feature that needs to be at the node.
It selects the node based on the level of importance of the feature.

For example, As we can see the education number provides one of the best insights into predicting the class.
The higher the education, the more probability is to have that person income >50K USD.

Methodology

For Methodology, We have followed a workflow given below for better evaluation and accuracy.

Workflow :
1. Preprocessing of Dataset
2. Comparison of initial classification models
3. Hyper-parameter optimization for the best-performed models.
4. Feature selection and comparison of accuracies with top 3 models

Preprocessing of Dataset :

1.1 Dealing with Outliers

As our Dataset has categorical values, There can't be any outliers as such.

To detect outlier, we have to define a distance metric between the features and datapoints
If that is the case, we can't do that as the values belong in the same category.
One example of this is for one feature- Marital status, Can we define the distance between Single and Married or Single and widowed? which is of little value or importance.

1.2 Dealing with Missing Values

For dealing with missing values :
We have replaced the missing values (Blanks and special characters) to Null (np. Nan)
Now, there are two approaches,
we can either drop the missing rows as its a small percentage of features 0.05 and 0.01)
or we can impute the values with mean of data or most common which is more plausible.
I have implemented both approaches but went forward with imputation.
So for numerical features such as age, hours per week, education number we have used simple Imputer with strategy as mean, whereas for categorical features (work-class, native country, occupation, etc.) we have used most common occurrences (Simple Imputer with strategy as most_frequent)

1.3 Handling Categorical Data

As our dataset has categorical data, We have implemented both Ordinal encoder and One-hot encoder.
Even though the accuracy for Ordinal encoder is more, It is most likely due to overfitting as 90% of our data is nominal.
So we have used a One-hot encoder for further processing.
Also, we have also manually mapped some of the features as the features had some noise in them.
For Example, For Marital Status, There are multiple values: Separated, Never married, Widowed which can be clustered into a single value - Single

1.4 Handling Imbalance

For handling the imbalance in data, first, we will check the imbalances using the confusion matrix and then use the SMOTE algorithm.

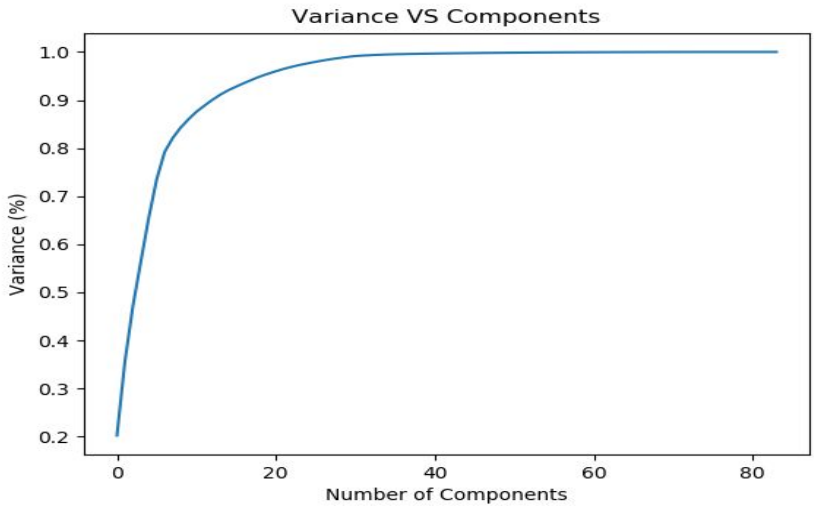
```
Original Accuracy :  0.8113420002047292
Confusion matrix:
 [[6474  936]
 [ 907 1452]]
Accuracy after balancing :  0.8058143105742656
```

1.5 Feature Selection

For feature selection, I have implemented 3 methods:
1. Univariate Feature Selection
1.1 Select K best
1.2 Select Percentile
2. Low variance removal
3. Greedy Feature Selection

1.6 Dimensionality Reduction

For dimensionality reduction, We have used PCA (Principal Component Analysis)
Ideally, we should use NCA (Neighborhood component Analysis) over PCA as our data is supervised, But due to Memory restriction, We have taken the PCA technique for dimensionality reduction.



For the Number of components, We have compared the variance in data and taken n_components value as 35 as the variance afterward is pretty much steady.

Models used for the Initial Model building phase :

1. **Gaussian Naive Bayes**
2. **K-Neighbors Classifier**
3. **Decision Tree Classifier**
4. **Random Forest Classifier**
5. **Logistic Regression**
6. **Linear Discriminant Analysis**
7. **AdaBoost Classifier**
8. **Quadratic Discriminant Analysis**
9. **MLP Classifier (Neural Nets)**
10. **Gradient Boosting Classifier**
11. **Extra Trees Classifier**
12. **XGBOOST Classifier**

Out of these models, Top 3 models which provided the best accuracy are :
Extra Trees - 0.8893810679611651
Random Forest - 0.8740088996763754
KNN - 0.8566545307443365

Hyper-parameter Optimization

For hyperparameter tuning, We have used **RandomizedSearchCV** to reduce computation power.

Due to memory restriction, I was not able to explore all the parameters, but I have stated the possible search space for maximum accuracy.

Parameter grid for Extra Trees :

```
paramgrid_ETC = {
   # "loss": ["deviance"],
   # "learning_rate": [0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2],
   # "min_samples_split": list(range(1, 10)),
   # "min_samples_leaf": list(range(1, 10)),
   "max_depth": list(range(2, 60, 2)),
   "max_features": [None, "log2", "sqrt"],
   # "criterion": ["friedman_mse", "mae"],
   "n_estimators": list(range(100, 600, 100)),
}
```

Ideal Parameters :

```
Best parameters for Extra Trees : {'n_estimators': 300, 'max_features':
'sqrt', 'max_depth': 40}
```

Parameter grid for Random Forest classifier :

```
paramgrid_RFC = {
       # "loss": ["deviance"],
       # "learning_rate": [0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2],
       # "min_samples_split": list(range(1, 10)),
       # "min_samples_leaf": list(range(1, 10)),
       "max_depth": list(range(2, 60, 2)),
       "max_features": [None, "log2", "sqrt"],
       # "criterion": ["friedman_mse", "mae"],
       "n_estimators": list(range(100, 500, 50)),
   }
```

Ideal Parameters :

```
Best parameters for Random Forest : {'n_estimators': 195, 'max_features':
'sqrt', 'max_depth': 56}
```

Parameter Explanation for ETC and RFC :

Max depth - The maximum depth for the tree to be explored
Max features - the number of features to consider for the best split
Estimators - The number of trees in the forest

Parameter grid for KNN :

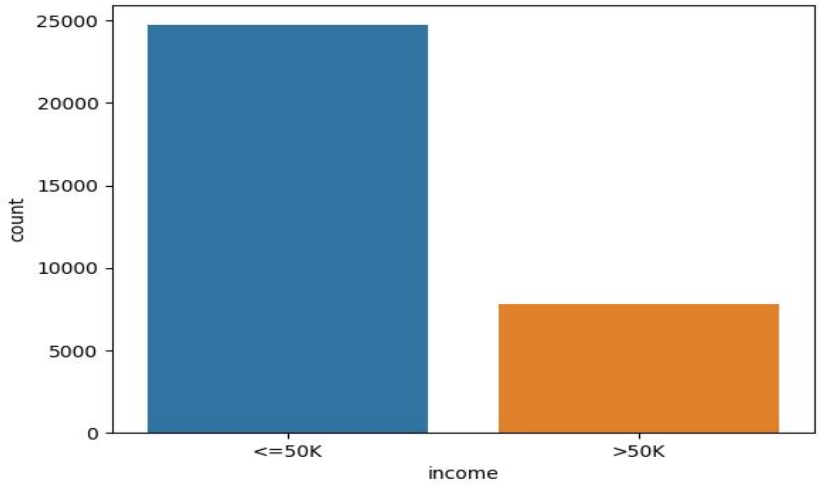
```
paramgrid_KNN = {
    # "algorithm": "auto",
    "n_neighbors": list(range(1, 50)),
    "weights": ["uniform", "distance"],
}
```

Ideal Parameters :

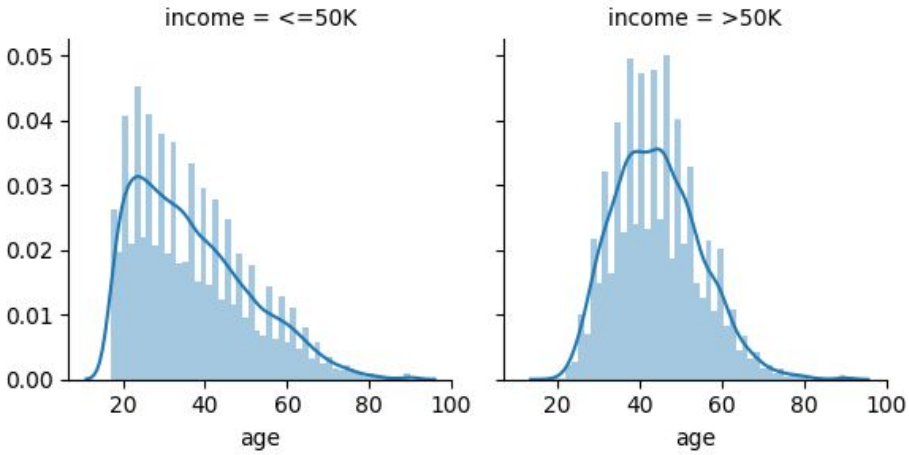
```
Best parameters for KNN : {'weights': 'uniform', 'n_neighbors': 1}
```

Evaluation and Conclusions

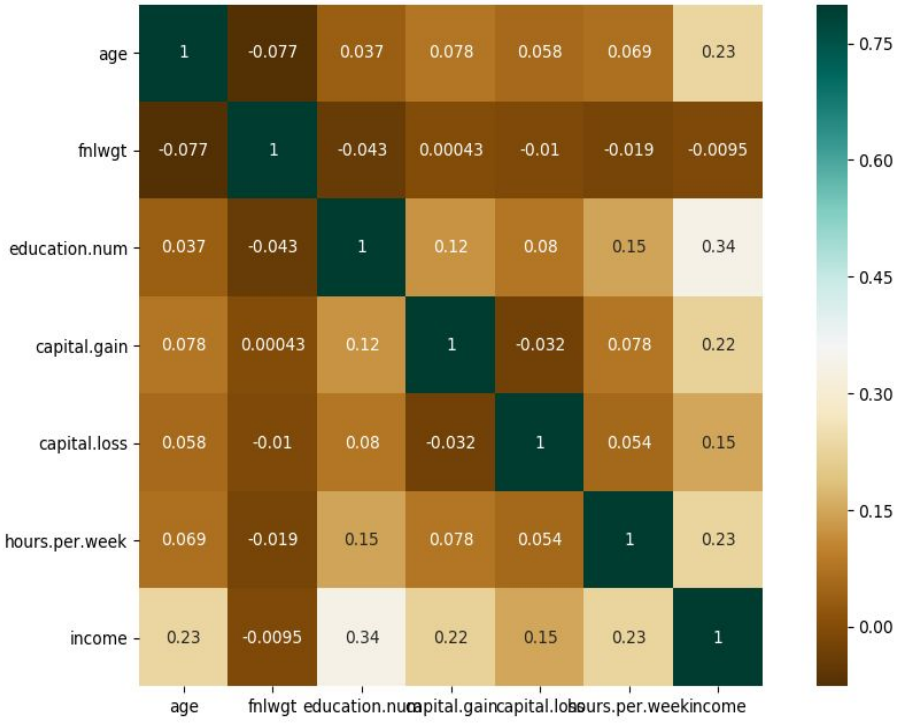
Class distribution :



Age VS Income (Label):

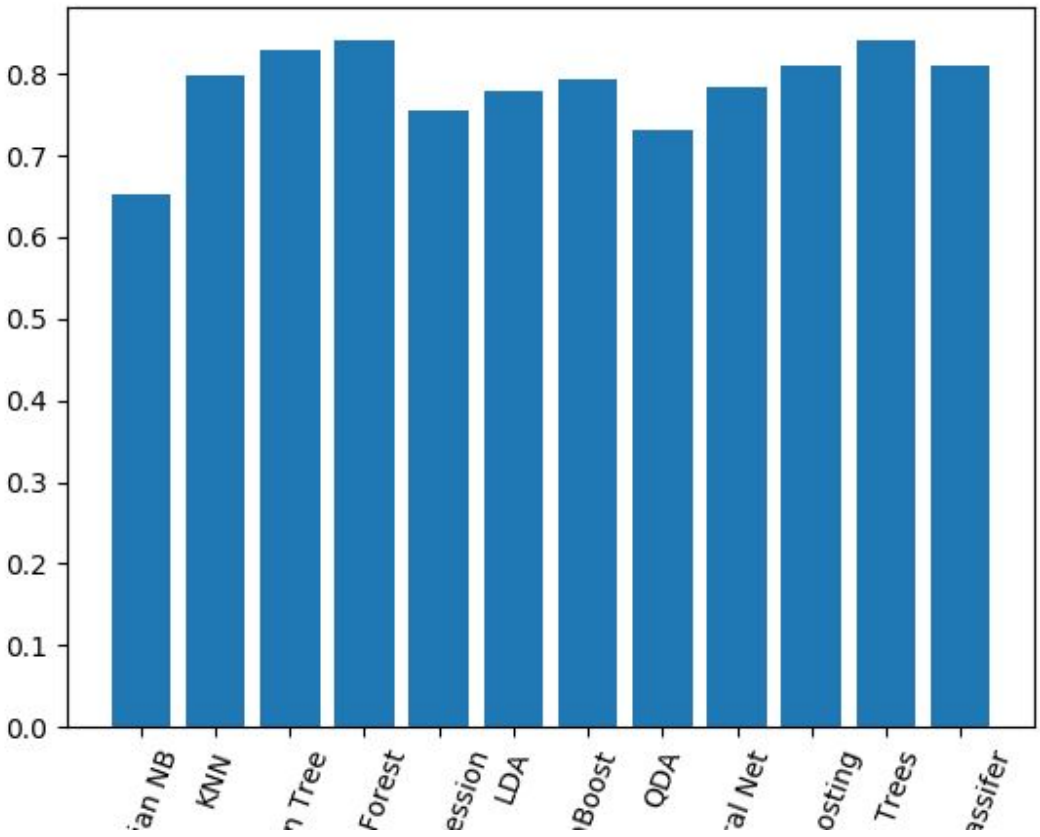


Relation Between Features :



Model Comparison :

Comparison of Classification Models with their accuracy.



Top 3 models :

```
### Classifier : Extra Trees  ###
Accuracy : 0.8756270226537216
precision : 0.8669948091026992
Recall : 0.895509708737864
F1 Score : 0.8786211328201128

 ### Classifier : Random Forest  ###
Accuracy : 0.8626011326860841
precision : 0.8551820817662648
```

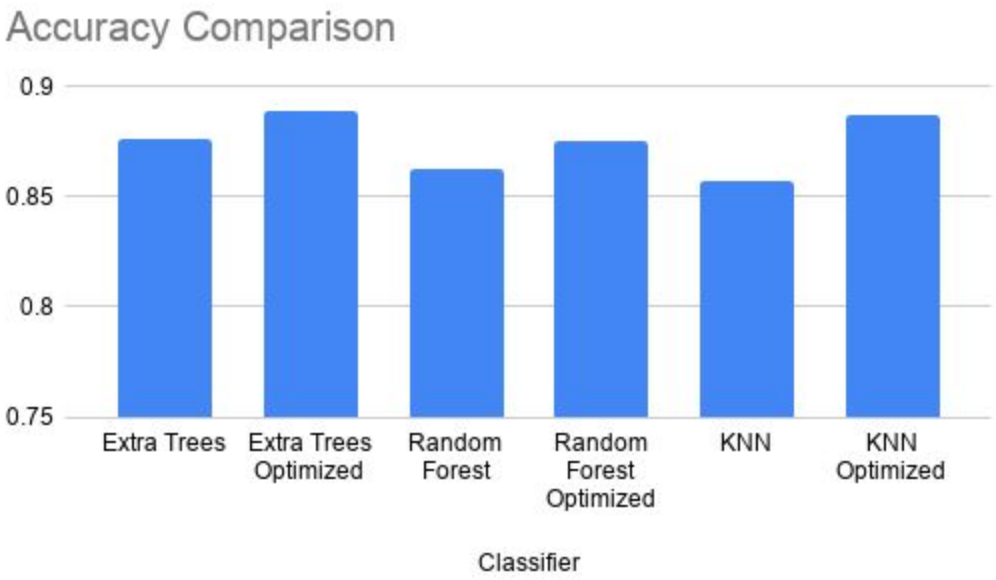
```
Recall : 0.8875
F1 Score : 0.8663835371035749

 ### Classifier : KNN  ###
Accuracy :  0.8567556634304208
precision :  0.8069422072157171
Recall :  0.94081715210356
F1 Score :  0.8683786165452745
```

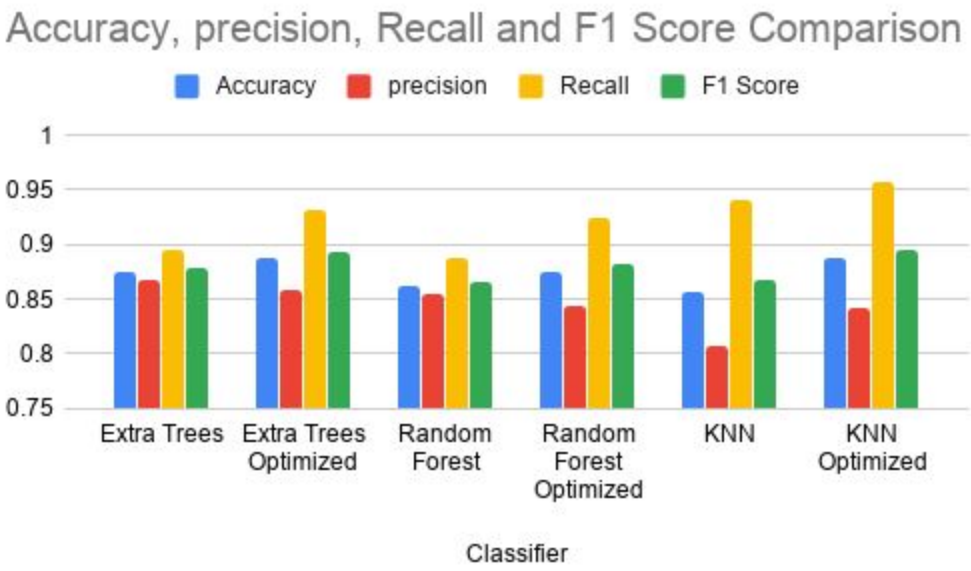
Hyper-parameter Tuning and Comparison with Optimized models :

Classifier	Extra Trees	Extra Trees Optimized	Increase in Accuracy
Accuracy	87.56%	88.85%	1.29%
precision	86.70%	85.83%	-0.87%
Recall	89.55%	93.23%	3.68%
F1 Score	87.86%	89.37%	1.51%

Classifier	Random Forest	Random Forest Optimized	Increase in Accuracy
Accuracy	86.26%	87.47%	1.21%
precision	85.52%	84.45%	-1.07%
Recall	88.75%	92.51%	3.76%
F1 Score	86.64%	88.19%	1.56%

Classifier	KNN	KNN Optimized	Increase in Accuracy
Accuracy	85.68%	88.72%	3.04%
precision	80.69%	84.15%	3.46%
Recall	94.08%	95.69%	1.61%
F1 Score	86.84%	89.51%	2.67%





As we can see from the graph and table, for Tree algorithms such as Random Forest and Extra trees, even though the accuracy is not increased dramatically (**1-2 %**), The recall seems to have improved, which means the number of identified true Positives is increased significantly. It suggests that the quality of classifiers is increased. Whereas, For K nearest neighbors, All the evaluation factors (Accuracy, Precision, Recall, and F1) seems to have improved.

Evaluation of Feature Selection :

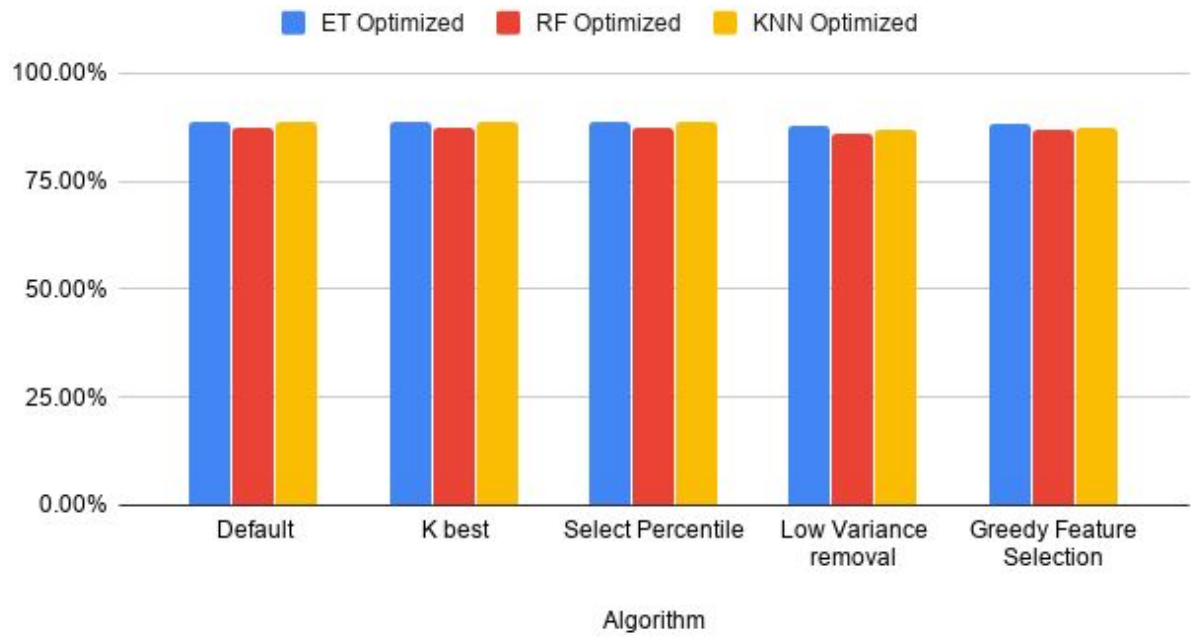
Algorithm	Scoring	Extra Trees	ET Optimized	Random Forest	RF Optimized	KNN	KNN Optimized
K best	Accuracy	87.50%	88.76%	86.10%	87.27%	85.64%	88.73%
K best	precision	86.27%	85.83%	84.72%	84.39%	80.68%	84.20%
K best	Recall	89.62%	93.27%	88.35%	92.54%	94.00%	95.66%
K best	F1Score	87.86%	89.35%	86.71%	88.21%	86.79%	89.52%

Algorithm	Scoring	Extra Trees	ET Optimized	Random Forest	RF Optimized	KNN	KNN Optimized
Select Percentile	Accuracy	87.78%	88.81%	85.99%	87.42%	85.63%	88.73%
Select Percentile	precision	85.94%	85.82%	84.62%	84.37%	80.65%	84.17%
Select Percentile	Recall	89.81%	93.33%	88.58%	92.50%	94.03%	95.70%
Select Percentile	F1 Score	87.89%	89.26%	86.36%	88.08%	86.79%	89.52%

Algorithm	Scoring	Extra Trees	ET Optimized	Random Forest	RF Optimized	KNN	KNN Optimized
Low Variance removal	Accuracy	86.77%	87.64%	84.85%	86.22%	84.51%	87.07%
Low Variance removal	precision	84.92%	84.30%	83.42%	82.87%	80.43%	82.91%
Low Variance removal	Recall	89.52%	93.05%	87.94%	91.93%	91.56%	93.75%
Low Variance removal	F1 Score	87.09%	88.37%	85.36%	87.16%	85.58%	87.94%

Algorithm	Scoring	Extra Trees	ET Optimized	Random Forest	RF Optimized	KNN	KNN Optimized
Greedy Feature Selection	Accuracy	87.42%	88.40%	85.62%	87.15%	85.09%	87.46%
Greedy Feature Selection	precision	86.16%	85.46%	84.48%	84.24%	80.91%	83.63%
Greedy Feature Selection	Recall	89.68%	92.98%	88.19%	92.25%	92.24%	93.54%
Greedy Feature Selection	F1 Score	87.74%	89.06%	86.14%	87.91%	86.14%	88.25%

Comparison of Feature Selection techniques :



As we can see, The application of different selection techniques didn't increase the accuracy of the overall model, In my opinion, that is due to the fact that the features were encoded and heavily reduced in dimensions (During PCA), So we are not able to see any drastic change in accuracy metric.

Conclusion :

In the prediction of the Adult Census Income dataset, We have found that the randomized tree algorithms work better with this dataset (Extra trees and Random Forest). This is due to the fact that we have a large number of numeric features and reduced computation to find optimal cut points and smoothening of data.
There are many feature selection and model improvement techniques left due to time and memory restriction as training the dataset was computationally expensive (Upto hours at a time) which could be implemented in the future.
For instance, we could use NCA for dimensionality reduction and perform feature selection with correlations before encoding the data to further improve the overall accuracy of the model.