

Income Prediction Using Machine Learning

Module Code: CMP7161

Module Coordinator: Yevgeniya Kovalcuk

Student Name: Muhammad Waleed

Student Id: 18157492

Submitted on: Date: 12th May, 2020

**Table of Content**

1. **Abstract**
2. **Aim**
3. **Introduction**
4. **Background**
5. **About Dataset**
6. **Problem Statement**
7. **Summary of Approach**

1. **Models**

**8.1. Logistic Regression**

**8.2. Decision Tree Classifier**

**8.3. Random Forest Classifier**

1. **Models Comparison**
2. **Conclusion**
3. **Reference**
4. **Abstract:**

This paper looks at the possibility of accurately predicting income salary brackets of individuals using three machine learning models trained with a census dataset. The algorithms used are the Logistic Regression classifier, Random Forest Classifier and the Decisions Tree Classifier. Using data cleaning, visualization and pre-processing techniques to improve the performance of the machine learning models, predictions were ultimately accurate with varying degrees of reliability and training time.

.

1. **Aim:**

The aim of this report is to identify whether it is possible, using census data, to predict the income bracket of individuals earning over or under $50k. The objectives of this paper are as follows:

* Identify whether it is possible for the prediction of income brackets using census data.
* Compare & contrast three machine learning models trained.
* Provide recommendations based on my findings for future research.

1. **Introduction:**

This report will investigate the capability of machine learning models in predicting individual income brackets, based on a census dataset. By training three distinct machine learning models. I will investigate the differences between each one, primarily with regards to accuracy and time taken to train. as they are incredibly important in ascertaining the viability of a model for a project.

In this report, I will use number of different supervised algorithms to precisely predict the income of individuals using Machine learning.

The three models trained are the Gaussian Naive Bayes classifier, Random Forest Classifier and the Decisions Tree Classifier, which attempt to categorize input data into a given set of categories, which in this case would be whether they are or are not earning over $50k.

1. **Background:**

Machine learning is a branch of artificial intelligence, aiming to give computers the ability to learn and thus behave with intelligence. Methods developed with this machine learning paradigm aim to “improve their performance at certain tasks based on observed data” (Ghahramani, 2015). Machine learning algorithms were first developed in the 1950s and 1960s, where statistical methods, symbolic learning and neural networks were at the forefront of data analysis (Kononenko, 2001). Before then, machine learning existed only in mathematical models, without the aid of computational algorithms to process large datasets. As decades of research have gone by, machine learning has become more sophisticated with the abstraction of complex functions, opening the accessibility of these techniques to users of varying degrees of mathematical capability. AI is a sub-zone of machine learning, whereby the term alludes to the capacity of IT frameworks to autonomously discover answers for issues by perceiving designs in databases. Machine Learning empowers IT frameworks to perceive designs based on existing calculations and informational collections and to create sufficient arrangement ideas.

In this manner, in Machine Learning, Artificial information is produced based on understanding. Machine learning is a strategy for information and data analysis that robotizes investigative model structure. as a part of AI is dependent on the possibility that frameworks can gain from information, distinguish examples, and settle on choices with negligible human intercession.

We can characterize Machine learning algorithms as follows:

* **Supervised machine learning algorithms**

It can apply on what has been realized in the past to new information utilizing marked guides to foresee future occasions.

* **Unsupervised machine learning algorithms**

These algorithms are utilized when the data used to prepare is neither grouped nor marked.

* **Semi-supervised machine learning algorithms**

This category falls in the middle of supervised and unsupervised learning since they utilize both marked and unlabeled information for preparing and normally a limited quantity of named information and a lot of unlabeled information

* **Reinforcement machine learning algorithms**

This is a learning technique that cooperates with its condition by delivering activities and finds errors or rewards.

**5. About Dataset:**

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)). The prediction task is to determine whether a person makes over $50K a year.

Description of fnlwgt (final weight):

The weights on the Current Population Survey (CPS) files are controlled to independent estimates of the civilian non-institutional population of the US. These are prepared monthly for us by Population Division here at the Census Bureau. We use 3 sets of controls. These are:

* A single cell estimate of the population 16+ for each state.
* Controls for Hispanic Origin by age and sex.
* Controls by Race, age and sex.

We use all three sets of controls in our weighting program and "rake" through them 6 times so that by the end we come back to all the controls we used. The term estimate refers to population totals derived from CPS by creating "weighted tallies" of any specified socioeconomic characteristics of the population. People with similar demographic characteristics should have similar weights. There is one important caveat to remember about this statement. That is that since the CPS sample is actually a collection of 51 state samples, each with its own probability of selection, the statement only applies within state.

Attribute Information:

1. Age
2. Work Class
3. Final Weights(fnlwgt)
4. Education
5. Education Number
6. Martial Status
7. Occupation
8. Relationship
9. Race
10. Sex

1. **Problem statement:**

You have been provided with an Excel dataset that has 15 columns and 48842 rows. Our task is to analyze the dataset and predict whether the income of an adult will exceed 50k per year or not by developing a supervised machine learning model.

1. **Summary of Approaches:**

The summary gives a brief description of the implementation of both approaches and modeling with the comparison. It is divided into six steps:

* Load Libraries
* Load Data
* Analyze Data
* Feature Engineering
* Modeling
* Finalizing the Model
  1. **Load Libraries:**

Followings are some libraries that are required for this project:

* Pandas
* Matplotlib
* Seaborn
* Sklearn

**[input]:**

**import** pandas **as** pd  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**import** seaborn **as** sns  
**from** sklearn.ensemble **import** RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, ExtraTreesClassifier, VotingClassifier  
**from** sklearn.discriminant\_analysis **import** LinearDiscriminantAnalysis  
**from** sklearn.linear\_model **import** LogisticRegression  
**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn.tree **import** DecisionTreeClassifier  
**from** sklearn.neural\_network **import** MLPClassifier  
**from** sklearn.naive\_bayes **import** GaussianNB  
**from** sklearn.ensemble **import** RandomForestClassifier  
**from** sklearn.model\_selection **import** GridSearchCV, cross\_val\_score, StratifiedKFold, learning\_curve, train\_test\_split, KFold  
**from** sklearn.metrics **import** classification\_report  
**from** sklearn.metrics **import** confusion\_matrix  
**from** sklearn.metrics **import** accuracy\_score

**7.2 Load Data:**

Now load the data from CSV file using pandas.

**[input]:**

dataset = pd.read\_csv(**"data.csv"**)  
  
*# Check for Null Data*dataset.isnull().sum()  
  
  
  
*# Replace All Null Data in NaN*dataset = dataset.fillna(np.nan)  
  
  
*# Get data types*dataset.dtypes  
  
*# Peek at data*dataset.head(4)  
  
*# Reformat Column We Are Predicting*dataset[**'income'**]=dataset[**'income'**].map({**'<=50K'**: 0, **'>50K'**: 1, **'<=50K.'**: 0, **'>50K.'**: 1})  
dataset.head(4)

**[output]:**

age int64

workclass object

fnlwgt int64

education object

education.num int64

marital.status object

occupation object

relationship object

race object

sex object

capital.gain int64

capital.loss int64

hours.per.week int64

native.country object

income object

dtype: object

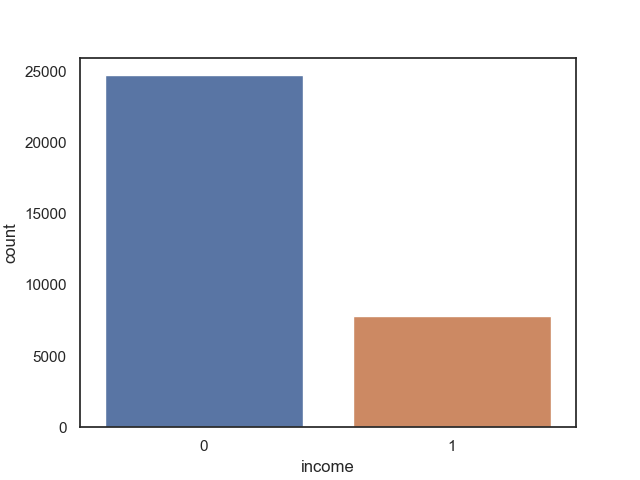
**7.3 Analyze the data:**

Now we will Analyze the gathered data

**[input]:**

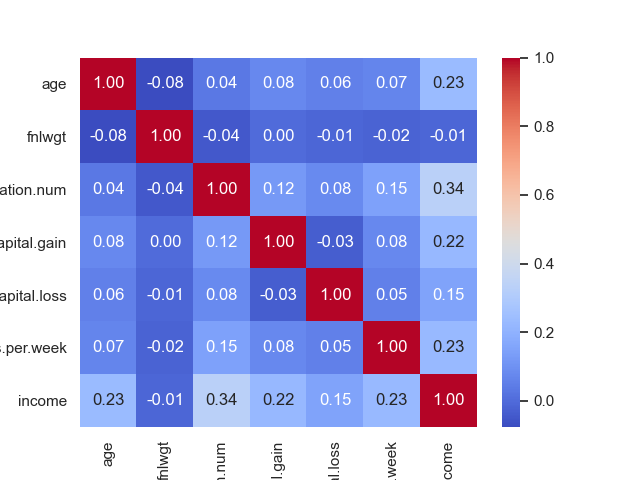
*# Identify Numeric features*numeric\_features = [**'age'**,**'fnlwgt'**,**'education.num'**,**'capital.gain'**,**'capital.loss'**,**'hours.per.week'**,**'income'**]  
  
*# Identify Categorical features*cat\_features = [**'workclass'**,**'education'**,**'marital.status'**, **'occupation'**, **'relationship'**, **'race'**, **'sex'**, **'native'**]  
  
*# Count of >50K & <=50K*sns.countplot(dataset[**'income'**],label=**"Count"**)  
plt.show()

**[output]:**



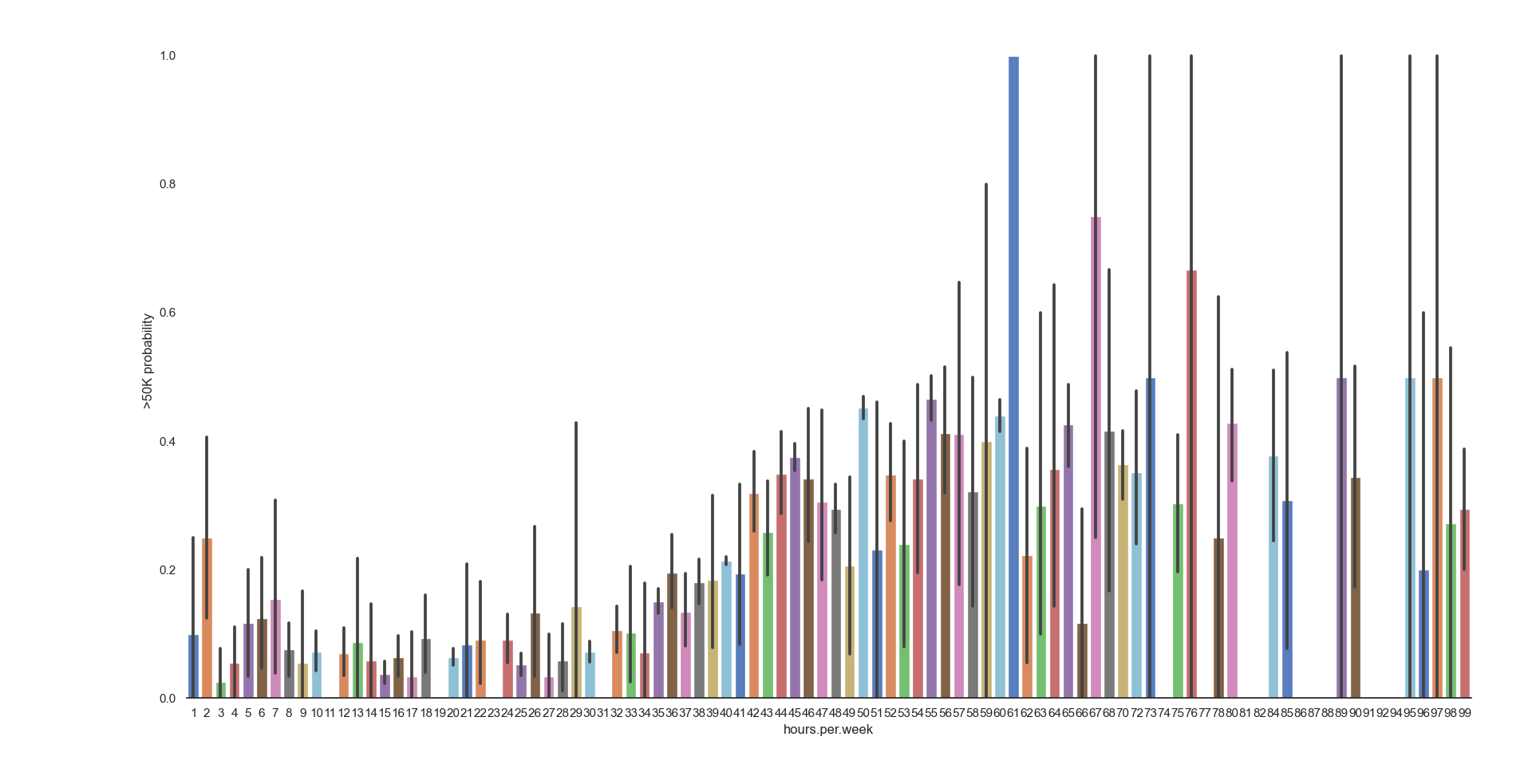
*# Correlation matrix between numerical values*g = sns.heatmap(dataset[numeric\_features].corr(),annot=**True**, fmt = **".2f"**, cmap = **"coolwarm"**)  
plt.show()

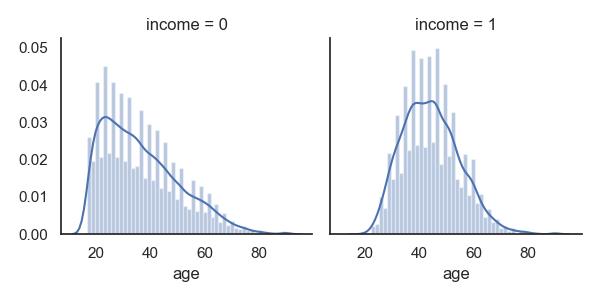
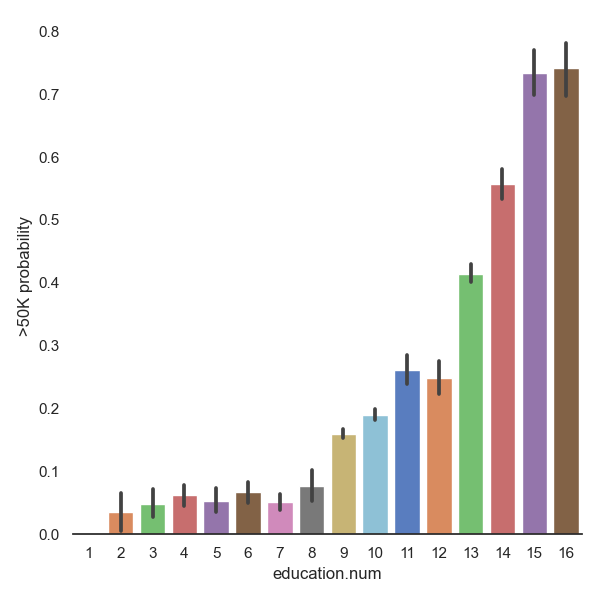
**[output]:**



*# Explore Education Num vs Income*g = sns.factorplot(x=**"education.num"**,y=**"income"**,data=dataset,kind=**"bar"**,size = 6,palette = **"muted"**)  
g.despine(left=**True**)  
g = g.set\_ylabels(**">50K probability"**)  
  
*# Explore Hours Per Week vs Income*g = sns.factorplot(x=**"hours.per.week"**,y=**"income"**,data=dataset,kind=**"bar"**,size = 6,palette = **"muted"**)  
g.despine(left=**True**)  
g = g.set\_ylabels(**">50K probability"**)  
  
*# Explore Age vs Income*g = sns.FacetGrid(dataset, col=**'income'**)  
g = g.map(sns.distplot, **"age"**)  
plt.show()

**[output]:**

****



*# Fill Missing Category Entries*dataset[**"workclass"**] = dataset[**"workclass"**].fillna(**"X"**)  
dataset[**"occupation"**] = dataset[**"occupation"**].fillna(**"X"**)  
dataset[**"native.country"**] = dataset[**"native.country"**].fillna(**"United-States"**)  
  
*# Confirm All Missing Data is Handled*dataset.isnull().sum()  
  
*# Fill Missing Category Entries*dataset[**"workclass"**] = dataset[**"workclass"**].fillna(**"X"**)  
dataset[**"occupation"**] = dataset[**"occupation"**].fillna(**"X"**)  
dataset[**"native.country"**] = dataset[**"native.country"**].fillna(**"United-States"**)  
  
*# Confirm All Missing Data is Handled*dataset.isnull().sum()

**[output]:**

Age 0

Workclass 0

Fnlwgt 0

Education 0

education.num 0

marital.status 0

Occupation 0

Relationship 0

Race 0

Sex 0

capital.gain 0

capital.loss 0

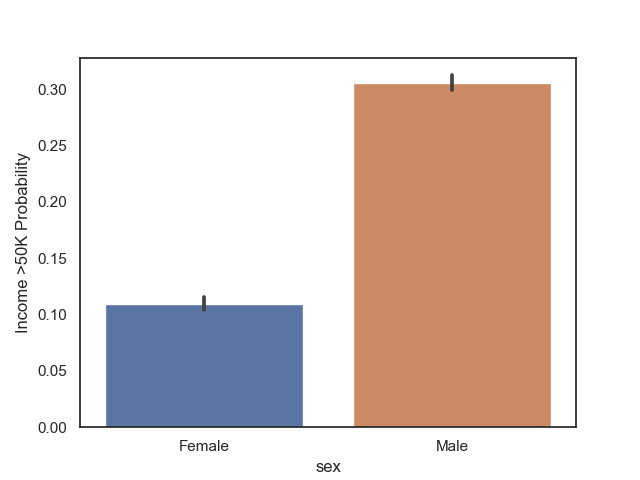
hours.per.week 0

native.country 0

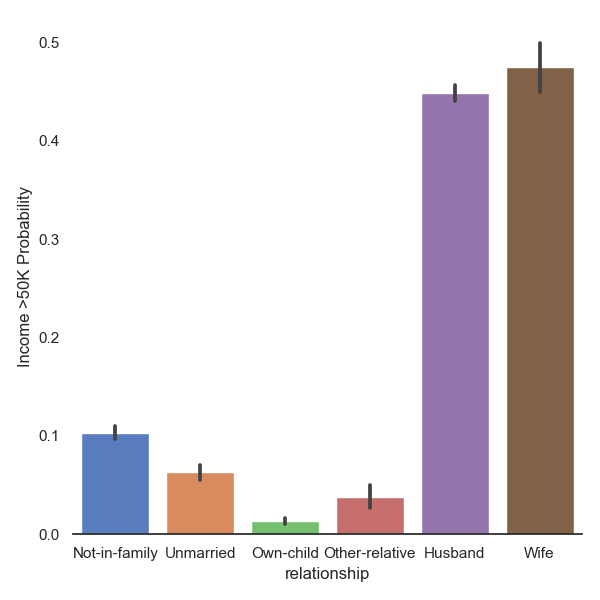
Income 0

dtype: int64  
  
*# Explore Sex vs Income*g = sns.barplot(x=**"sex"**,y=**"income"**,data=dataset)  
g = g.set\_ylabel(**"Income >50K Probability"**)  
plt.show()

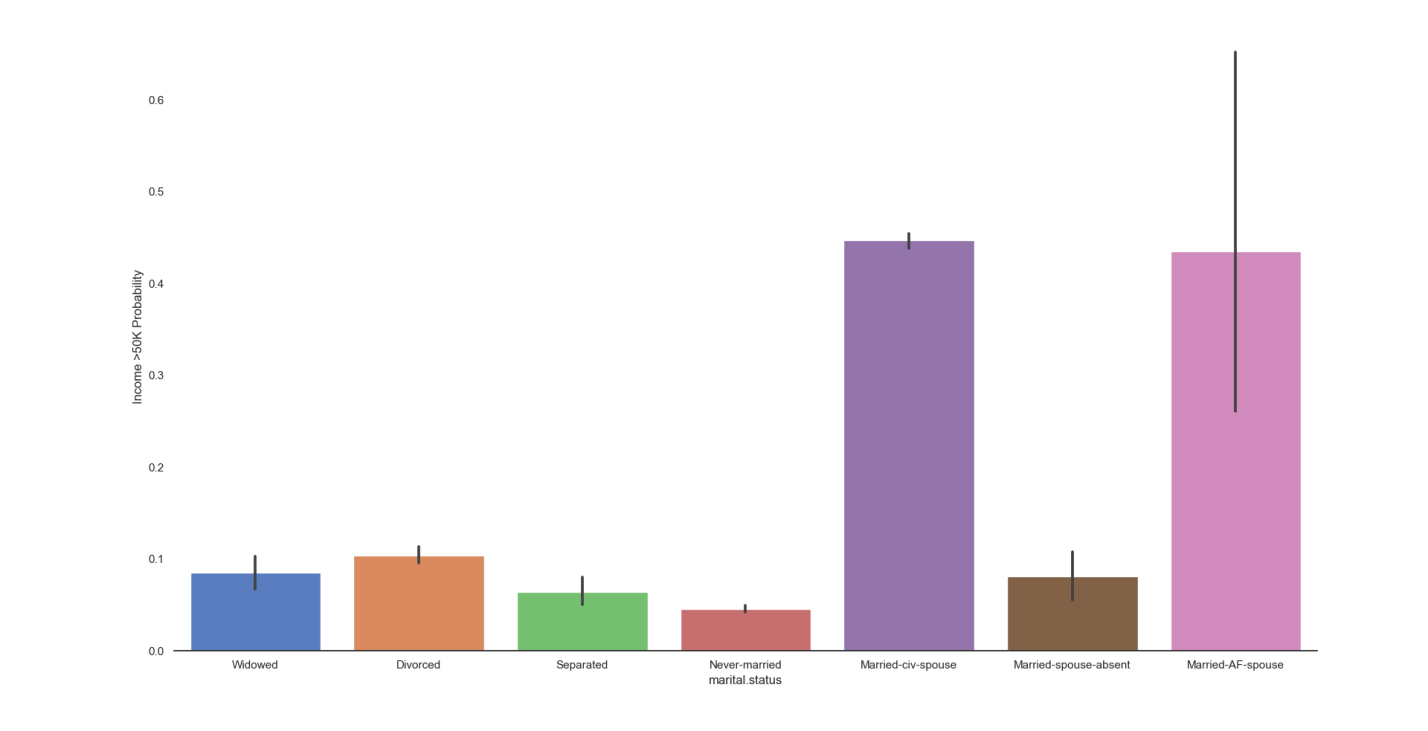
**[output]:**

  
  
*# Explore Relationship vs Income*g = sns.factorplot(x=**"relationship"**,y=**"income"**,data=dataset,kind=**"bar"**, size = 6 ,  
palette = **"muted"**)  
g.despine(left=**True**)  
g = g.set\_ylabels(**"Income >50K Probability"**)  
plt.show()

**[output]:**

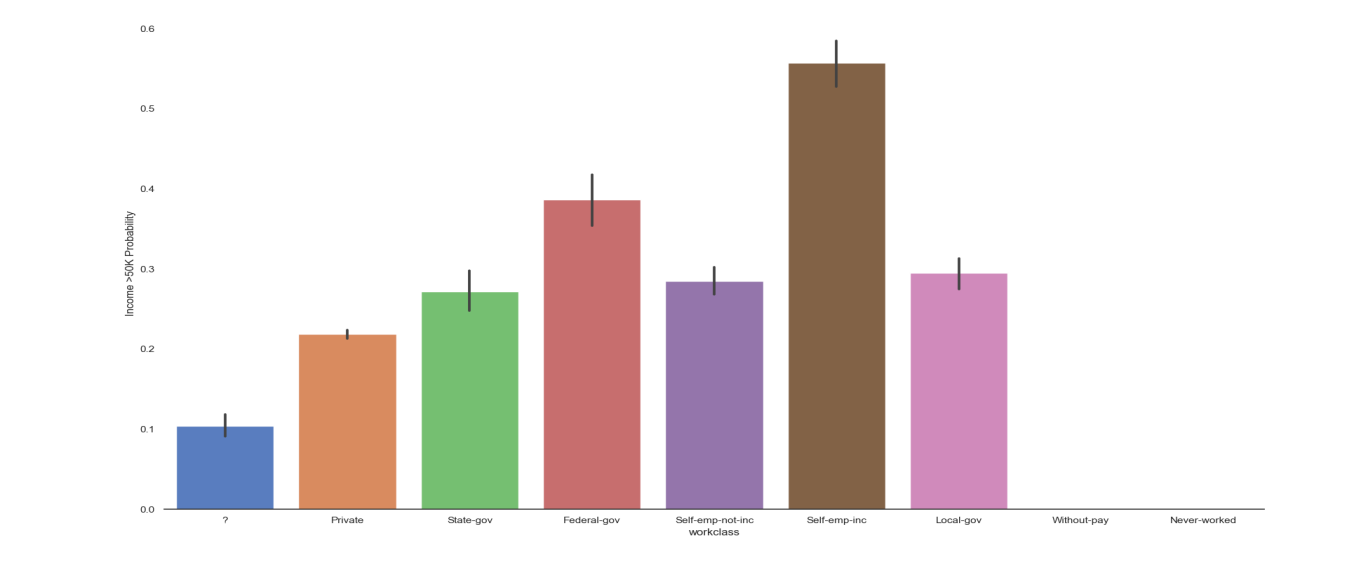
  
  
*# Explore Marital Status vs Income*g = sns.factorplot(x=**"marital.status"**,y=**"income"**,data=dataset,kind=**"bar"**, size = 6 ,  
palette = **"muted"**)  
g.despine(left=**True**)  
g = g.set\_ylabels(**"Income >50K Probability"**)  
plt.show()

**[output]:**



*# Explore Workclass vs Income*g = sns.factorplot(x=**"workclass"**,y=**"income"**,data=dataset,kind=**"bar"**, size = 6 ,  
palette = **"muted"**)  
g.despine(left=**True**)  
g = g.set\_ylabels(**"Income >50K Probability"**)  
plt.show()

**[output]:**



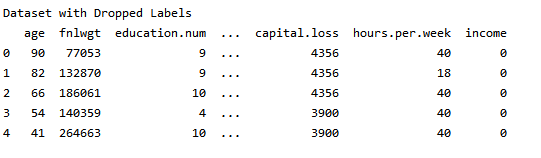
**7.4 Feature Engineering:**

Now we will do feature scaling of the data set so that our models should give good precision and accurate results,

**[input]:**

*# Convert Sex value to 0 and 1*dataset[**"sex"**] = dataset[**"sex"**].map({**"Male"**: 0, **"Female"**:1})  
  
*# Create Married Column - Binary Yes(1) or No(0)*dataset[**"marital.status"**] = dataset[**"marital.status"**].replace([**'Never-married'**,**'Divorced'**,**'Separated'**,**'Widowed'**], **'Single'**)  
dataset[**"marital.status"**] = dataset[**"marital.status"**].replace([**'Married-civ-spouse'**,**'Married-spouse-absent'**,**'Married-AF-spouse'**], **'Married'**)  
dataset[**"marital.status"**] = dataset[**"marital.status"**].map({**"Married"**:1, **"Single"**:0})  
dataset[**"marital.status"**] = dataset[**"marital.status"**].astype(int)  
  
*# Drop the data you don't want to use*dataset.drop(labels=[**"workclass"**,**"education"**,**"occupation"**,**"relationship"**,**"race"**,**"native.country"**], axis = 1, inplace = **True**)  
print(**'Dataset with Dropped Labels'**)  
print(dataset.head())

**[output]:**

****

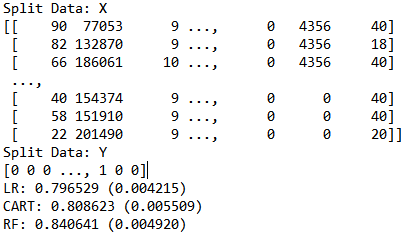
**7.5 Modelling:**

Now we will create models that will predict our future results

**[input]:**

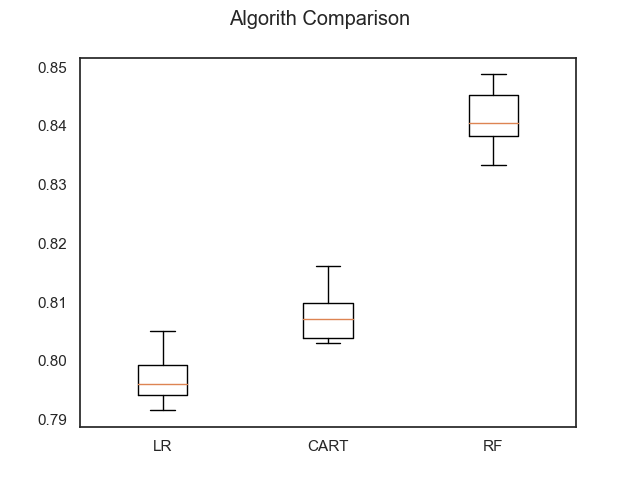
*# Split-out Validation Dataset and Create Test Variables*array = dataset.values  
X = array[:,0:8]  
Y = array[:,8]  
print(**'Split Data: X'**)  
print(X)  
print(**'Split Data: Y'**)  
print(Y)

**[output]:**



validation\_size = 0.20  
seed = 7  
num\_folds = 10  
scoring = **'accuracy'**X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X,Y,  
 test\_size=validation\_size,random\_state=seed)  
  
*# Params for Random Forest*num\_trees = 100  
max\_features = 3  
  
*#Spot Check 5 Algorithms (LR, LDA, KNN, CART, GNB, SVM)*models = []  
models.append((**'LR'**, LogisticRegression()))  
models.append((**'CART'**, DecisionTreeClassifier()))  
models.append((**'RF'**, RandomForestClassifier(n\_estimators=num\_trees, max\_features=max\_features)))  
*#models.append(('SVM', SVC()))  
# evalutate each model in turn*results = []  
names = []  
**for** name, model **in** models:  
 kfold = KFold(n\_splits=10, random\_state=seed)  
 cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring=**'accuracy'**)  
 results.append(cv\_results)  
 names.append(name)  
 msg = **"%s: %f (%f)"** % (name, cv\_results.mean(), cv\_results.std())  
 print(msg)  
  
fig = plt.figure()  
fig.suptitle(**'Algorith Comparison'**)  
ax = fig.add\_subplot(111)  
plt.boxplot(results)  
ax.set\_xticklabels(names)  
plt.show()

**[output]:**



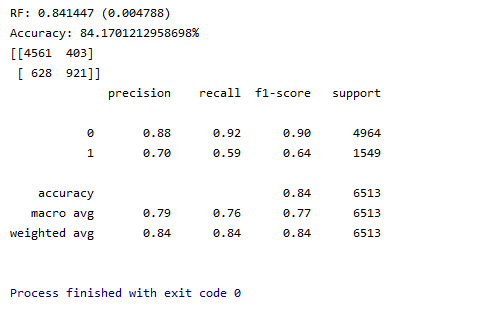
**7.6 Finalizing Model:**

Finalizing the best model from out trained model that give the best and accurate results.

**[input]:**

random\_forest = RandomForestClassifier(n\_estimators=250,max\_features=5)  
random\_forest.fit(X\_train, Y\_train)  
predictions = random\_forest.predict(X\_validation)  
print(**"Accuracy: %s%%"** % (100\*accuracy\_score(Y\_validation, predictions)))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))

**[output]:**



1. **Models:**

Following are the models used:

**8.1. Logistics Regression:**

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. logistic regression models the probabilities for characterization issues with two potential results. It's an augmentation of the linear regression model for arrangement and classification issues. In contrast of linear regression that outcomes continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. Logistic regression is the proper regression analysiss to direct when the reliant variable is binary. Like all regression investigations and analysis, the logistic regression is a prescient analysis.Logistic regression is utilized to depict information and to clarify the connection between one dependent binary variable and one or more independent binary variable. Logistic regression (LR) is a measurable technique like direct or linear regression since LR finds a condition that predicts a result for a double factor Y from at least one reaction factors X.

Logistic regression has been broadly utilized by a wide range of individuals, however it battles with its prohibitive expressiveness and different models may have better prescient execution.

Followings are the types of logistic regression:

* Binary (e.g. True or False) It gives two outcomes.
* Multi-linear functions fail Class (e.g. Cats, dogs or Sheep's)
* Ordinal Logistic Regression

We will use Binary Logistics Regression to perform analysis.

We used sklearn’s Python Library to implement this Model and get the predictions.

**[input]:**

Showing you the basic code where we appended this sklearn library to use it in our model

models.append((**'LR'**, LogisticRegression()))

**[output]:**

LR: 0.796836 (0.003727)

Its Prediction Accuracy is 79%

**8.2. Decision Tree Classifier:**

Decision Tree Analysis is a general, predictive modeling tool that has applications spanning several different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Decision tree classifier lies in category of **Supervised machine learning,** where the information is ceaselessly part as indicated by a specific boundary.

It consisits of **nodes, branch and leaf nodes.**

**Nodes** tests for the estimation of a specific trait.

**Branch:** it correspond to the result of a test and associate with the following node or leaf.

**Leaf nodes** is the terminal node that foresee the result.

There are two main types of Decision trees:

* Regression trees
* Classification trees

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

Decision Tree Classifier is a basic and broadly utilized grouping procedure. It applies a straightforward thought to tackle the grouping issue. decision Tree Classifier represents a progression of painstakingly made inquiries concerning the properties of the test record. Each time it get an answer, a subsequent inquiry is posed until a decision about the calss name of the record is reached.

We used sklearn’s Python Library to implement this Model and get the predictions.

**[input]:**

Showing you the basic code where we appended this sklearn library to use it in our model

models.append((**'CART'**, DecisionTreeClassifier()))

**[output]:**

CART: 0.807663 (0.004307)

Its Prediction Accuracy is 80%

**8.3. Random Forest Classifier:**

Random forest is a supervised learning algorithm that is used for both classifications as well as regression. However, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees mean more robust forests. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution using voting. A random forest classifier is an estimator that fits various decision tree classifiers on different sub-tests of the dataset and utilizations averaging to improve the prescient exactness.

Random Forest algorithms keeps up great precision even a huge extent of the information is absent. It beats the issue of overfitting by averaging or consolidating the consequences of various choice trees. It **functions** admirably for an enormous scope of information things than a solitary decision tree does.

Random forest is an adaptable, simple to utilize machine learning algorithm that produces, even without hyper-boundary tuning, an incredible outcome more often than not.

Random Forest Classifier consists of troupe or essemble calculation. Ensembled calculations are those which consolidates more than one calculations of same or distinctive kind for grouping objects Random forest classifier makes a lot of decision trees from arbitrarily chosen subset of preparing set. It at that point totals the votes from various decision trees to choose the last class of the test object. It is an ensemble method that is better than a single decision tree because it reduces the over-fitting by averaging the result.

How it works:

* Pick at random K data points from the training set.
* Build the decision tree associated with those K data points.
* Choose the number n tree of trees you want to build and repeat steps 1 & 2.
* For a new data point, make each one of your n tree trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values.

We used sklearn’s Python Library to implement this Model and get the predictions.

**[input]:**

Showing you the basic code where we appended this sklearn library to use it in our model

models.append((**'RF'**, RandomForestClassifier(n\_estimators=num\_trees, max\_features=max\_features)))

**[output]:**

RF: 0.841063 (0.004807)

Its Prediction Accuracy is 84%

1. **Model Comparison:**

Followings points shows the comparison between models:

* Logistic Regression with its Prediction Accuracy 79%
* Decision Tree Classifier with its Prediction Accuracy 80%
* Random Forest Classifier with its Prediction Accuracy 84%

From the above accuracy and metrics, the best performing model in the test data is the Random Forest Classifier, which has an accuracy score of 84%. So, I will prefer that model to Predict income of individuals if some data is given.

1. **Conclusion:**

Based on the results obtained and discussed within this paper, I believe that machine learning models can indeed predict income brackets based on census information about an individual, requiring far fewer features than those provided in the dataset used.

On the bases of the above representations and modeling approaches, it is clear that the random forest classifier gives the most accurate output. But other models also give approx 80% accurate prediction that are useful too

Bearing in mind the time to train, my recommendation for binary classification tasks such as this would be to use the random forest classifier algorithm, providing exceptional performance and accuracy. To further improve upon reliability of predictions, I would also recommend using the cross-validation technique to train each machine learning model, rather than performing a single train/test split as carried out in this paper.

1. **Reference:**

Followings are the reference which helped us in this project

1. Dataset available at <https://www.kaggle.com/ipbyrne/income-prediction-84-369-accuracy/data?select=adult.csv>
2. Documentation for library used: <https://scikit-learn.org/stable/user_guide.html>
3. Support: http://libanswers.bcu.ac.uk/