# INCOME ANALYSIS OF CENSUS DATABASE USING PYSPARK

*Abhishek R*
*abhishekravichandran23@gmail.com*

*Pavan Dutt*
*1832040mds@cit.edu.in*

*Sahana S R*
*sahanamscds2018@gmail.com*

*3rd Year, MSc. Data Science (Integrated),*
*Coimbatore Institute of Technology, Coimbatore*

*Dr.M.Sujithra M.C.A,M.Phil,Ph.D,*
*sujithra@cit.edu.in*

*Dr.P.Velvadivu M.C.A.,M.Phil.,Ph.D.*
*velvadivu@cit.edu.in*

*Assistant Professor, Coimbatore Institute of Technology, Coimbatore*

## Abstract:

The data set contains information about every individual's age, education level and various other features from the census along with the income . The income feature is a categorical feature with two classes i.e., less than 50k dollars or greater than 50k dollars. The problem is to build a machine learning model that could effectively predict the income of people given the input features. Considering the size of the data and the dimension of the data, the model is built using Big Data Techniques.

## Data set source:

This data was extracted from the census bureau database found at https://archive.ics.uci.edu/ml/datasets/adult

## Data set Attributes:

- **age**: continuous.
- **workclass**: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- **fnlwgt**: continuous. The weights on the CPS files are controlled to independent estimates of the civilian noninstitutional 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 socio-economic 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.
- 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.
- education-num: continuous.
- marital-status: Married-civ-spouse, Divorced, Never-married, Separated,

Widowed, Married-spouse-absent, Married-AF-spouse.

- 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.
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex: Female, Male.
- capital-gain: continuous.
- capital-loss: continuous.
- hours-per-week: continuous.
- 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.
- income:>50K, <=50K.

## Dataset Sample:

| age | workclass | fnlwgt | education | education_ | marital_status | occupation | relationship | race | sex | capital_gain | capital_loss | hours_per_ | native_country | income |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White | Male | 2174 | 0 | 40 | United-States | <=50K |
| 50 | Self-emp-n | 83311 | Bachelors | 13 | Married-civ-spo | Exec-manage | Husband | White | Male | 0 | 0 | 13 | United-States | <=50K |
| 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-clea | Not-in-family | White | Male | 0 | 0 | 40 | United-States | <=50K |
| 53 | Private | 234721 | 11th | 7 | Married-civ-spo | Handlers-clea | Husband | Black | Male | 0 | 0 | 40 | United-States | <=50K |
| 28 | Private | 338409 | Bachelors | 13 | Married-civ-spo | Prof-specialty | Wife | Black | Female | 0 | 0 | 40 | Cuba | <=50K |
| 37 | Private | 284582 | Masters | 14 | Married-civ-spo | Exec-manage | Wife | White | Female | 0 | 0 | 40 | United-States | <=50K |
| 49 | Private | 160187 | 9th | 5 | Married-spouse | Other-service | Not-in-family | Black | Female | 0 | 0 | 16 | Jamaica | <=50K |
| 52 | Self-emp-n | 209642 | HS-grad | 9 | Married-civ-spo | Exec-manage | Husband | White | Male | 0 | 0 | 45 | United-States | >50K |
| 31 | Private | 45781 | Masters | 14 | Never-married | Prof-specialty | Not-in-family | White | Female | 14084 | 0 | 50 | United-States | >50K |

*Fig 1*

## Tools Used:

- Python
- Apache Spark MLlib
- Apache Spark SQL
- Pyspark
- Hive

## Techniques and Methodology:

### Creation of Spark Session and reading the data:

A spark session is firstly initialized. The entry point into all functionality in Spark is the SparkSession class. SparkSession in Spark 2.0 provides builtin support for Hive features including the ability to write queries using HiveQL, access to Hive UDFs, and the ability to read data from Hive tables. To use these features, we do not need to have an existing Hive setup. After creating the spark session, the data is read in the form of RDD ( Resilient Distributed Dataset). Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster.

```
root
 |-- age: integer (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: integer (nullable = true)
 |-- education: string (nullable = true)
 |-- education_num: integer (nullable = true)
 |-- marital_status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital_gain: integer (nullable = true)
 |-- capital_loss: integer (nullable = true)
 |-- hours_per_week: integer (nullable = true)
 |-- native_country: string (nullable = true)
 |-- income: string (nullable = true)
```

*Fig 2*

### Descriptive statistics of the data:

The descriptive statistics include the mean, median , standard deviation , count, minimum and maximum values for each attribute.

| summary | age | workclass | fnlwgt | education | education_num | marital_status | occupati | lationship | race | sex | capital_gain | capital_loss | hours_per_week | native_country | income |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 32561 | 32561 | 32561 | 32561 | 32561 | 32561 | 3256 | 32561 | 32561 | 32561 | 32561 | 32561 | 32561 | 32561 | 32561 |
| mean | 38.58164675532078 | null | 189778.36651208502 | null | 10.0806793403151 | null | nul | null | null | null | 1077.6488437087312 | 87.303829734959 | 40.437455852092995 | null | null |
| stddev | 13.640432553581356 | null | 105549.97769702227 | null | 2.572720332067397 | null | nul | null | null | null | 7385.292084840354 | 402.960218649002 | 12.347428681731838 | null | null |
| min | 17 | Federal-gov | 12285 | 10th | 1 | Divorced | Adm-clerica | Husband | Amer-Indian-Eskimo | Female | 0 | 0 | 1 | Cambodia | <=50K |
| max | 90 | Without-pay | 1484705 | Some-college | 16 | Widowed | Transport-movir | Wife | White | Male | 99999 | 4356 | 99 | Yugoslavia | >50K |

*Fig 3*

## Exploratory Data Analysis and Data Understanding

Initially a view for the data is created. After performing various type of querying on the data, an overall picture of the data and the distribution of values in the data is obtained

| workclass | avg(hours_per_week) | avg(capital_loss) | avg(capital_gain) |
|---|---|---|---|
| State-gov | 39.03158705701079 | 83.25654853620955 | 701.6995377503852 |
| Federal-gov | 41.37916666666667 | 112.26875 | 833.2322916666667 |
| Self-emp-not-inc | 44.421881149153876 | 116.63164108618655 | 1886.0617866981504 |
| Local-gov | 40.98279980888677 | 109.85427615862399 | 880.202580028667 |
| Private | 39.64234469264634 | 78.56815587803685 | 868.0810370128811 |
| Self-emp-inc | 48.81810035842294 | 155.13888888888889 | 4875.693548387097 |
| Without-pay | 32.714285714285715 | 0.0 | 487.85714285714283 |
| Never-worked | 28.428571428571427 | 0.0 | 0.0 |

*Fig 4*

From fig 4 we can see that Self employed people who tend to spend more working hours a week have less capital loss and more capital gain people who work in Local Govt tend to have more capital loss for their capital gain

| income | avg(hours_per_week) | avg(capital_loss) | avg(capital_gain) |
|---|---|---|---|
| >50K | 45.473026399693914 | 195.00153041703865 | 4006.142456319347 |
| <=50K | 38.840210355987054 | 53.14292071197411 | 148.75246763754046 |

*Fig 5*

From fig 5 we can see that people who earn more than 50k dollars tend to work longer and there is a huge difference in the average capital gains of people who earn less than 50k dollars and greater than 50k dollars per year.

| occupation | avg(hours_per_week) | avg(capital_gain) | avg(capital_loss) |
|---|---|---|---|
| Farming-fishing | 46.989939637826964 | 589.7263581488934 | 63.07545271629779 |
| Handlers-cleaners | 37.947445255474456 | 257.5729927007299 | 45.635766423357666 |
| Prof-specialty | 39.15828179842888 | 2072.9755975263247 | 112.84857095102791 |
| Adm-clerical | 37.55835543766578 | 495.9549071618037 | 60.794429708222815 |
| Exec-managerial | 44.9877029021151 | 2262.7729955730447 | 138.83841613379244 |
| Craft-repair | 42.30422054159551 | 649.5128080019517 | 88.46523542327397 |
| Sales | 40.78109589041096 | 1319.829315068493 | 98.30054794520548 |
| Tech-support | 39.432112068965516 | 673.5528017241379 | 98.66594827586206 |
| Transport-moving | 44.65623043206011 | 490.3237319974953 | 81.48090169067001 |
| Protective-serv | 42.87057010785824 | 708.0986132511556 | 78.33436055469954 |
| Armed-Forces | 40.666666666666664 | 0.0 | 209.66666666666666 |
| Machine-op-inspct | 40.755744255744254 | 328.6893106893107 | 61.70629370629371 |
| Other-service | 34.70166919575114 | 191.30166919575115 | 38.25068285280729 |
| Priv-house-serv | 32.88590604026846 | 279.8523489932886 | 21.449664429530202 |

*Fig 6*

From fig 6 we can see that people who work in farming and fishing tend to work for more hours in a week and yet don't get capital gain as much as people who work in other sectors. Similarly people who work in transportation sector also don't get paid as much as people from higher sectors in spite of working for more than 46 hours a week.

| sex | avg(hours_per_week) | avg(capital_gain) | avg(capital_loss) |
|---|---|---|---|
| Male | 42.42808627810923 | 1329.3700780174393 | 100.21330885727397 |
| Female | 36.410361154953115 | 568.4105468387336 | 61.18763346021725 |

*Fig 7*

Fig 7 tells us that the average capital gain of males is very high when compared to that of females.

| education | avg(hours_per_week) | avg(capital_gain) | avg(capital_loss) |
|---|---|---|---|
| Prof-school | 47.425347222222222 | 10414.416666666666 | 231.203125 |
| 10th | 37.052518756698824 | 404.57449088960345 | 56.845659163987136 |
| 7th-8th | 39.36687306501548 | 233.93962848297213 | 65.6687306501548 |
| 5th-6th | 38.8978978978979 | 176.02102102102103 | 68.25225225225225 |
| Assoc-acdm | 40.504217432052485 | 640.3992502343018 | 93.41893158388004 |
| Assoc-voc | 41.61070911722142 | 715.0513748191028 | 72.75470332850941 |
| Masters | 43.83633197910621 | 2562.563551944283 | 166.71967498549043 |
| 12th | 35.78060046189376 | 284.0877598152425 | 32.33718244803695 |
| Preschool | 36.64705882352941 | 898.3921568627451 | 66.49019607843137 |
| 9th | 38.04474708171206 | 342.08949416342415 | 28.99805447470817 |
| Bachelors | 42.614005602240894 | 1756.299533146592 | 118.35032679738562 |
| Doctorate | 46.973365617433416 | 4770.145278450364 | 262.8450363196126 |
| HS-grad | 40.575373773926295 | 576.800114274831 | 70.46662222645462 |
| 11th | 33.92595744680851 | 215.09787234042554 | 50.07914893617021 |
| Some-college | 38.85228363736113 | 598.8241667809629 | 71.63708681936635 |
| 1st-4th | 38.25595238095238 | 125.875 | 48.32738095238095 |

*Fig 8*

Fig 8 tells us that people with higher education, tend to work less and yet gain more income while people with poor education tend to work longer hours and yet could not earn much.

## Checking for missing values:

| age | workclass | fnlwgt | education | education_num | marital_status | occupation | relationship | race | sex | capital_gain | capital_loss | hours_per_week | native_country | income |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Fig 9*

From fig 9 we can see that that we thankfully don't have any missing values to be handled

## Distinct categories in each attribute:

| workclass | education | marital_status | occupation | relationship | race | native_country |
|---|---|---|---|---|---|---|
| 8 | 16 | 7 | 14 | 6 | 5 | 41 |

*Fig 10*

Fig 10 tells us the number of distinct classes in each feature

## String Indexing:

StringIndexer encodes a string column of labels to a column of label indices. StringIndexer can encode multiple columns.

## One Hot Encoding:

One-hot encoding maps a categorical feature, represented as a label index, to a binary vector with at most a single one-value indicating the presence of a specific feature value from among the set of all feature values. This encoding allows algorithms which expect continuous features, such as Logistic Regression, to use categorical features. For string type input data, it is common to encode categorical features using StringIndexer first. OneHotEncoder can transform multiple columns, returning an one-hot-encoded output vector column for each input column. It is common to merge these vectors into a single feature vector using VectorAssembler.

## Vector Assembling:

VectorAssembler is a transformer that combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees. VectorAssembler accepts the following input column types: all numeric types, boolean type, and vector type. In each row, the values of the input columns will be concatenated into a vector in the specified order.

| features | income_Index |
|---|---|
| (97,[3,7,8,11,24,... | 0.0 |
| (97,[1,7,8,11,24,... | 0.0 |
| (97,[0,7,8,9,24,2... | 0.0 |
| (97,[0,7,8,14,24,... | 0.0 |
| (97,[0,7,8,11,24,... | 0.0 |
| (97,[0,7,8,12,24,... | 0.0 |
| (97,[0,7,8,19,24,... | 0.0 |
| (97,[1,7,8,9,24,2... | 1.0 |
| (97,[0,7,8,12,24,... | 1.0 |
| (97,[0,7,8,11,24,... | 1.0 |
| (97,[0,7,8,10,24,... | 1.0 |
| (97,[3,7,8,11,24,... | 1.0 |
| (97,[0,7,8,11,24,... | 0.0 |
| (97,[0,7,8,15,24,... | 0.0 |
| (97,[0,7,8,13,24,... | 1.0 |
| (97,[0,7,8,17,24,... | 0.0 |
| (97,[1,7,8,9,24,2... | 0.0 |
| (97,[0,7,8,9,24,2... | 0.0 |
| (97,[0,7,8,14,24,... | 0.0 |
| (97,[1,7,8,12,24,... | 1.0 |

only showing top 20 rows

*Fig 11*

Fig 11 shows the final data after string indexing , one hot encoding and vector assembling.

## Standard Scaling:

StandardScaler transforms a dataset of Vector rows, normalizing each feature to have unit standard deviation and/or zero mean

| scaledFeatures |
|---|
| (97,[3,7,8,11,24,... |
| (97,[1,7,8,11,24,... |
| (97,[0,7,8,9,24,2... |
| (97,[0,7,8,14,24,... |
| (97,[0,7,8,11,24,... |
| (97,[0,7,8,12,24,... |
| (97,[0,7,8,19,24,... |
| (97,[1,7,8,9,24,2... |
| (97,[0,7,8,12,24,... |
| (97,[0,7,8,11,24,... |
| (97,[0,7,8,10,24,... |
| (97,[3,7,8,11,24,... |
| (97,[0,7,8,11,24,... |
| (97,[0,7,8,15,24,... |
| (97,[0,7,8,13,24,... |
| (97,[0,7,8,17,24,... |
| (97,[1,7,8,9,24,2... |
| (97,[0,7,8,9,24,2... |
| (97,[0,7,8,14,24,... |
| (97,[1,7,8,12,24,... |

only showing top 20 rows

*Fig 12*

Fig 12 shows the data after performing scaling

## Logistic regression:

Logistic regression is a popular method to predict a categorical response. It is a special case of Generalized Linear models that predicts the probability of the outcomes. In spark.ml logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression. After fitting logistic regression to the data, the following coefficients and intercepts are obtained.

```
Coefficients:
[-0.9824086758898924,-0.7419872626158511,-0.5696549640009633,-0.4872050015671399,-0.3622481980005578,-0.2931993894038852,-0.241
4035052402708,0.3211477188318392,0.05930411293728575,0.4741803933646044,0.3397726737672524,0.05916998270565335,-0.00278286133
0639354,0.112948572522943,0.2662560752687501,0.007707666570440325,0.2790233382910743,0.29791585059551207,0.002569840768920795,
0.2741688568854555,0.10113237033482995,-0.03818334571910119,0.2997229801375503,0.2425372619393289,1.3457251587502115,-0.4565859
177120672,-1.5133056229207147,-0.92623978777601,-0.48605925296667624,-0.45761778185378427,-0.28426503699603584,2.15041571612920
37,0.25517000712829485,0.39382396317983553,-0.19579704915140617,-0.21352510731089314,0.01918611464940211,-0.2319230629567449,-
0.13811599958808637,-0.46862212386863883,-0.21615167607286626,-0.1631074407606077,-0.27679077415186215,-0.29339477109512335,-0.
009100706718297167,-0.019754254171037884,-0.3125841957731438,0.3242494546879176,0.34339700850670357,-0.14558065466645317,0.2238
9028570909641,0.41545056470947805,0.11049633764186705,0.019549499836646722,0.023146226224615755,-0.025256327605481323,0.3829334
988352134,-1.4098228754218924,-0.8092447928658848,-0.3380203196813467,-0.30408255623831776,-0.29394378628681467,-0.322331065938
98675,-0.32400375348248506,-0.2893811285309121,-0.25780662613172534,-0.2640841125497276,-0.24140504664780713,-0.350161962079819
87,-0.291687554145503,-0.2212088725153151,-0.31406192947453787,-0.2949006836541981,-0.22936565900199934,-0.21258538741601787,-
0.21610468816738662,-0.30141027365751083,-0.1806600956841106,-0.19422767332862273,-0.20742819680274213,-0.17919454585772385,-0.
1920333823299268,-0.41579118195463366,-0.11165838688456994,-0.17929920568847135,-0.16583678331067314,-0.11924002132271996,-0.11
686439828803905,-0.09667224521829904,-0.12866287498397214,-0.12953922902594414,-0.15223059025265692,-0.08468432886822312,-0.282
03979916671357,-0.12361111082738158,-0.09101823148737492,-0.09549461858629082]
Intercept:
-2.1404319265187386
```

*Fig 13*

The training and test accuracy of logistic regression is as follows:

```
Training accuracy for Logistic Regression is 0.7653573049612654
Test accuracy for Logistic Regression is 0.7654543626196985
```

*Fig 14*

From the above output it is clear that the training accuracy is 76% and the test accuracy is also 76% . This is not an acceptable performance. This can be improved by using other algorithms

## Linear Support Vector Machines:

The linear SVM is a standard method for large-scale classification tasks. It is a linear method as described above in equation (1), with the loss function in the formulation given by the hinge loss:

$$L(w;x,y):=\max\{0,1-ywTx\}.$$

By default, linear SVMs are trained with an L2 regularization. We also support alternative L1 regularization. In this case, the problem becomes a linear program.

The linear SVMs algorithm outputs an SVM model. Given a new data point, denoted by x, the model makes predictions based on the value of wTx. By the default, if wTx≥0 then the outcome is positive, and negative otherwise.

The following are the coefficients and intercepts of linear SVM:

```
Coefficients:
[-0.0539894921279526З,-0.042739332053325335,-0.028621306157245063,-0.030755987528638645,0.05423134002998867,-0.002395048290037
922,-0.004461474675371362,-0.025822317041086278,-0.02409129725042768,-0.07431102024310013,-0.040829173116395626,0.0593452947423
9168,0.09059722755393178,-0.00872207151916139,-0.03627057921781257,-0.004870384598170359,-0.037425823713904065,-0.0469208851929
6775,0.10113587002890148,-0.031638873976989765,-0.020014774609857462,0.0923644872590346,-0.028427412523941025,-0.02177526466444
487,-0.019779236301192802,0.02751699882578852,-0.09520307116740967,-0.032005896974550266,-0.018486419477346683,-0.0156726280892
33257,-0.013920710266179828,0.19174011186534876,0.08348650123948233,-0.006031612901741015,0.003034721817812371,-0.0424044394549
1047,0.08292910717202932,-0.02575055753045966,-0.004101142632983752,-0.042789010725545674,-0.02951365937615953,-0.026108319739З
60335,-0.028699618337521177,-0.03137972735831251,0.0010295901251363949,-0.004433033665302738,-0.010118599528309466,0.0383769223
1770948,-0.05298717315800353,-0.05648649797520641,-0.039591897685278134,0.004414223469918464,-0.04765407662185999,-0.0405841683
3379731,-0.04504507523666937,-0.026345159323171302,0.0,-0.08625311550063491,-0.05451471519489705,-0.011279095849313664,-0.02103
145621200016,-0.01585557463454319,-0.027276879311128106,-0.02237295112886714,-0.01706901231344983,-0.011962722777157778,-0.0155
35699368659618,-0.015131250672340064,-0.0298196896306742,-0.021288734696556188,-0.01117399603840857,-0.021742095353526784,-0.02
1538704151854028,-0.01568877994427117,-0.005532819462612705,-0.013021509738810567,-0.023803157233500682,-0.005334174389801З315,
-0.016052914150596485,-0.010604209834502838,-0.013069021121182382,-0.013923259119598769,-0.010145599645204244,0.0,-0.0134042994
78616266,-0.011761950396212426,-0.007024767445548685,-0.005474986666287163,-0.004275500702543042,-0.012615056375288604,-0.00781
5971903635208,-0.0053039665674965465,4.244403387679115e-05,-0.013489528142338067,-0.007802707869687975,-0.005077497470690155,-
0.005802720742252855]
Intercept:
-0.044285844063469915
```

*Fig 15*

The training and test accuracy of linear SVM are as follows:

```
Training accuracy for Linear support vector classifier is 0.7653573049612654
Test accuracy for Linear support vector classifier is 0.7654543626196985
```

*Fig 16*

Linear SVM also does not tend to improve accuracy much.

## Decision Tree:

Decision trees are a popular family of classification and regression methods. More information about the spark.ml implementation can be found further in the section on decision trees.A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

The training and test accuracy of Decision Tree is below:

```
Training accuracy for decision tree is 0.839551
Test accuracy for decision tree is 0.83959
```

*Fig 17*

The performance of the decision tree is better than both logistic regression and linear SVM but this can still be improved.

## Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However,

data characteristics can affect their performance.
The training and test accuracy for Random forest is as follows:

```
Training accuracy for Random forest with 200 trees is 0.876151
Trest accuracy for Random forest with 200 trees is 0.853693
```

*Fig 18*

From fig 18 we can see that the training and test accuracy is much higher than the decision tree classifier. In order to obtain a better accuracy , Gradient-boosted tree classifier can be tried.

## Gradient-boosted tree classifier:

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems.

Gradient boosting is also known as gradient tree boosting, stochastic gradient boosting (an extension), and gradient boosting machines, or GBM for short.Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting.
The training and test accuracy of GBT is as follows:

```
Training accuracy for gradient boosted tree is 0.960141
Test accuracy for gradient boosted tree  0.829343
```

*Fig 18*

## Overall Inference:

After performing the above algorithms, we obtained the following accuracies

| Model | Training Accuracy | Test Accuracy |
|---|---|---|
| Logistic Regression | 76% | 76% |
| Linear SVM | 76% | 76% |
| Decision Tree | 83% | 83% |
| Random Forest | 87% | 85% |
| Gradient Boost Tree | 96% | 82% |

Logistic Regression and Linear SVM both did not perform well on both training and test set. Decision tree performed considerably well on both training and test set. Random forest also performed well on both training and test set and gave the highest test accuracy. Gradient boost tree performed extremely well on training data but did not do well on test data as much. This clearly indicates that the Gradient Boost tree is overfitting the training data. This is because Gradient Boost Tree is having a high variance problem. Hence the optimum model considering the bias variance trade-off is the Random forest model with 200 trees.

## Conclusion:

Hence an optimum model is found using bias variance trade-off analysis and the income of new entries is predicted with 85% confidence.

## Reference:


1. https://spark.apache.org/docs/latest/mllib-linear-methods.html#linear-support-vector-machines-svms
2. https://spark.apache.org/docs/latest/sql-getting-started.html
3. https://machinelearningmastery.com/gradient-boosting-machine-ensemble-in-python/#:~:text=Gradient%20boosting%20refers%20to%20a,machines%2C%20or%20GBM%20for%20short.
4. https://archive.ics.uci.edu/ml/datasets/Adult
5. https://spark.apache.org/docs/latest/api/python/index.html
6. https://en.wikipedia.org/wiki/Random_forest#:~:text=Random%20forests%20or%20random%20decision,average%20prediction%20(regression)%20of%20the