(7082CEM)

Coursework
Demonstration of a Big Data Program

MODULE LEADER: Dr. Marwan Fuad
Student Name: Manoj Sibi Mogan

SID: 10224061

INDIVIDUAL'S ANNUAL INCOME ANALYSIS THROUGH MACHINE LEARNING USING PYSPARK

I can confirm that all work submitted is my own: Yes

Table of Contents

1. Introduction ---------- 3

2. Implementation ---------- 4

2.1. Background study ---------- 4

2.1.1. SPARK ---------- 4

2.1.2. SparkSession ---------- 4

2.1.3. DataFrame ---------- 4

2.1.4. PySpark ---------- 4

2.2. Dataset ---------- 6

2.3. Installation Steps ---------- 7

2.4. Pre-processing the Dataset ---------- 10

2.5. Data Visualization ---------- 16

2.6. Machine Learning – Linear Regression ---------- 22

2.7. Discussion ---------- 26

2.8. Conclusion ---------- 27

2.9. References ---------- 28

3. Appendix ---------- 28

1. Introduction

In this paper I am going to explain and demonstrate how to setup and use PySpark along with the results of this study. There are many sophisticated platforms available which can handle massive amounts of data. Spark is one of the famous distributed computational platform for big data analysis. It has outstanding functionalities like performing faster in large dataset, was of use, fault tolerance and overcoming memory latency. Spark supports different programming languages. For this project I have chosen Python. Python has special functionalities like real time screen analytics, facilitates data visualization and process faster on the framework.

I have used PySpark which is a combination of Spark and Python in the study of Spark data processing which easily integrate and collaborate with RRD through navigating Py4J library. I have used Python, Spark and Machine learning (linear regression) to identify the correlation, accuracy and error rate along with visual data representation using Tableau. World-wide individual annual income data has been considered for this study. Due to the huge volume of data, I have extracted partial data from the Kaggle dataset and used for Big data analysis and data visualization. There are many parameters available, but I have selected Educational qualification, Age, Occupation and Income for my study.

Below are the steps the analysis was performed:

- PySpark was launched using number of installation steps.
- Pre-processing of the data set was done using loading data (Loading Dataframe).
- Duplicates were removed and columns were dropped including handling missing values.
- Dataframe operations were completed through different methods - Group-by, Distinct, Orderly, built-in functions, describe function and check value for specific columns.
- Following this was the exploratory analysis and data visualization is shown in Tableau.

2. Implementation

2.1. Background study

SPARK

Spark is a flexible in-memory framework which can handle batch and real-time analytic and data processing. Spark excels at the task that cause bottlenecks on disc IO in MapReduce. It is installed in JVM (Java Virtual Machine) along with the programming language Scala. It provides higher levels of abstraction than MapReduce and enhances developer productivity. Spark is referred to as a replacement to Hadoop, but in real-time Spark and other elements of the BDAS were designed to work closely with Hadoop HDFS and YARN, and several Spark implementations use HDFS for persistent storage.

In Spark, data is represented as resilient distributed datasets (RDD). RDDs are collections of objects that can be partitioned across 'n' number nodes of the cluster. The partitioning and subsequent distribution of processing are handled automatically by the Spark framework. The Spark API defines high-level methods that perform operations on RDDs. Operations such as joining, filtering, and aggregation, which would entail hundreds of lines of Java code in MapReduce, are expressed as simple method calls in Spark.

SPARKSESSION

Spark session is the entry point for the Spark SQL. It's the initial object to be created while developing a Spark SQL application. It allows for creating a DataFrame creating a Dataset, accessing the Spark SQL services, executing a SQL query, loading a table and accessing DataFrameReader interface to load a dataset of the format of the developer's choice.

DATAFRAME

Dataframe in PySpark is the distributed collection of structured or semi-structured data. This data in Dataframe is stored in rows under named columns which is similar to the relational database tables or excel sheets. It also shares some common attributes with RDD like Immutable in nature, follows lazy

evaluations and is distributed in nature. It supports a wide range of formats like JSON, CSV, TXT and many more.

PYSPARK

PySpark is a great language for performing exploratory data analysis at scale, building machine learning pipelines, and creating ETLs for a data platform. Spark is a popular open source framework that ensures data processing with lightning speed and supports various languages like Scala, Python, Java, and R. It then boils down to your language preference and scope of work. The goal of this post is to show how to get up and running with PySpark and to perform common tasks.

The diagram below shows how both PySpark and Spark contexts are managed by Spark driver with the aid of local file system and through communicating with Spark worker through cluster manager.

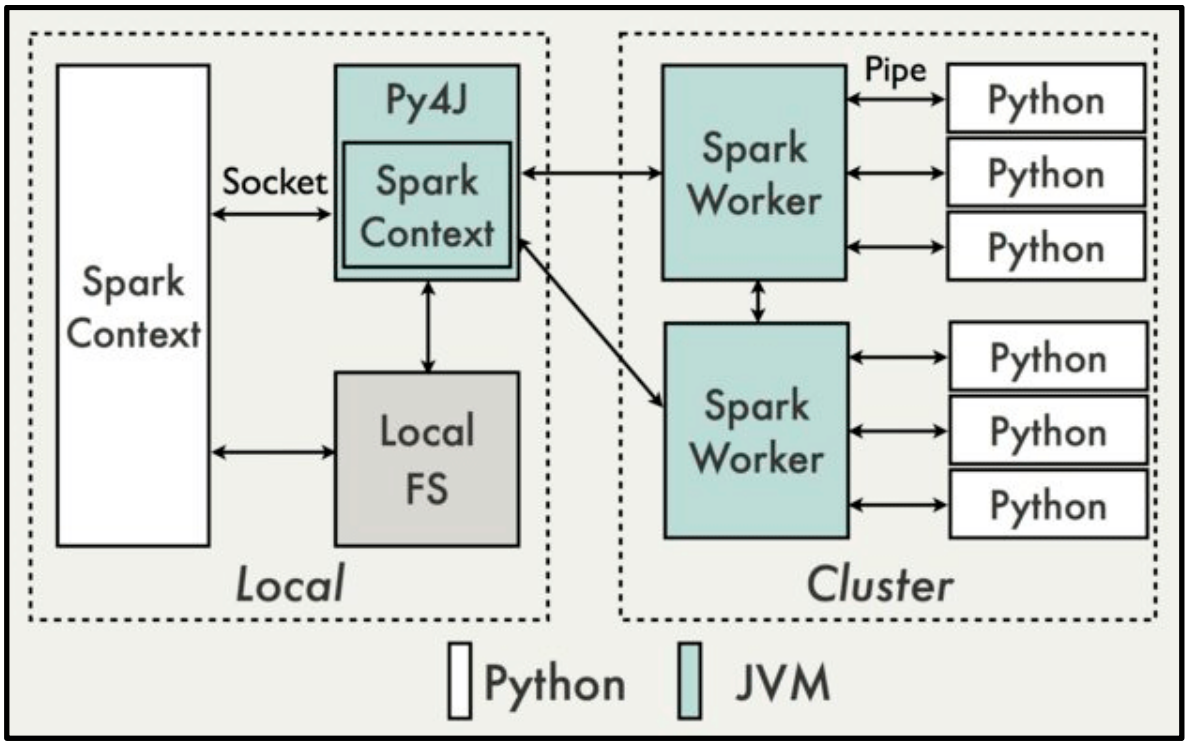

Image 1: PySpark Flow Diagram

2.2. Dataset

An individual's annual income results from various factors. Intuitively, it is influenced by the individual's education level, age, gender, occupation, and etc. This dataset which is from US Census Reports (1994 – 1996). Therefore, a version of this dataset has been extracted from Kaggle dataset and used in this study for Big Data analysis and Data visualisation. This chosen dataset includes information on Education Qualification, Age, Occupation, Hours Per Week, Income etc. and demographics. There are 13 attributes and contains 32, 561 number of records in this study. The main aim of this study is to analyse income status of different employees through analysing the factors such income, educational qualification, working sector etc.

The table below shows the attributes from the dataset including its description and data types.

Attribute	**Description**	**Data Types**
Age	Age of the employee	integer
Work Sector	Employee working sector	string
Educational_Qualification	Employee educational qualification details	string
Marital_Status	Employee Marital details	string
Occupation	Employee work nature	string
Relationship	Employee's relationship with their family	string
Country	Country to which the employee belongs	string
Income	Annual income of the employee	integer

2.3. Installation Steps

Machine used to perform this study has the hard disk drive with the capacity of 1TB, RAM with 16GB and the used operating system is macOS Catalina version 10.15.7. The steps installation steps followed are explained below:

Step 1: Initially, it is important to check whether java is installed within the machine through looking at the current java version by entering the below mentioned command in Terminal. If the java is installed it will display the below message.

```
(base) manoj@MANDYs-MacBook-Pro ~ % java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
(base) manoj@MANDYs-MacBook-Pro ~ %
```

Image 2.1: Step 1

And if the java is not installed below mentioned command needs to be used followed by the system password.

sudo apt install default-jdk

```
(base) manoj@MANDYs-MacBook-Pro ~ % sudo apt install default-jdk
Password:
```

Image 2.2: Step 1.1

Step 2:- Download the latest version Spark from (https://spark.apache.org/downloads.html) and save the zip folder in the desktop or any folder. Enter the below command in the terminal to unzip the Spark file.

tar -xzf spark-2.4.7-bin-hadoop2.7.tar

Step 3:- Under bash profile setting up the Spark Environment by using the command shown below.

```
export SPARK_HOME=/Users/manoj/desktop/softwares/spark-2.4.7-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin
```

Image 2.3: Step 3

Step 4:- To ensure whether the Spark is installed correctly with the below command.

spark-shell

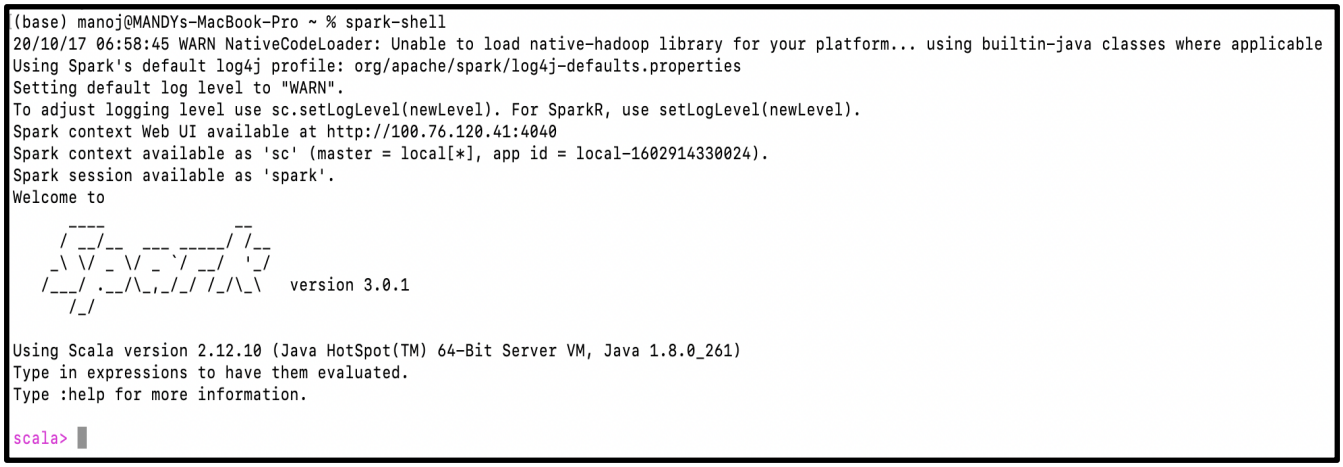

Image 2.4: Step 4

Step 5:- After the installation of Spark, need to install Python, Jupyter Notebook by downloading Anaconda (https://www.anaconda.com/products/individual#macos) and install it.

Step 6:- After the installation of Anaconda, it is important to check whether Python is installed in the machine. The following command will be used to check the Python version.

python -version

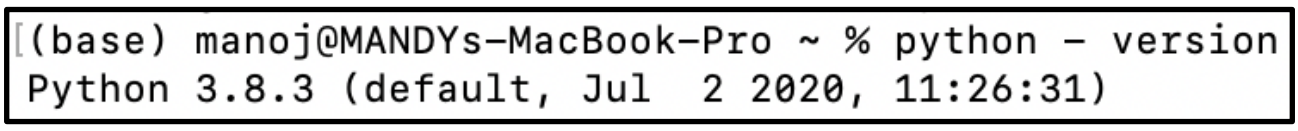

Image 2.5: Step 6

Step 7: After checking the Python version make sure whether Jupyter notebook is successfully installed the following command is used.

jupyter notebook

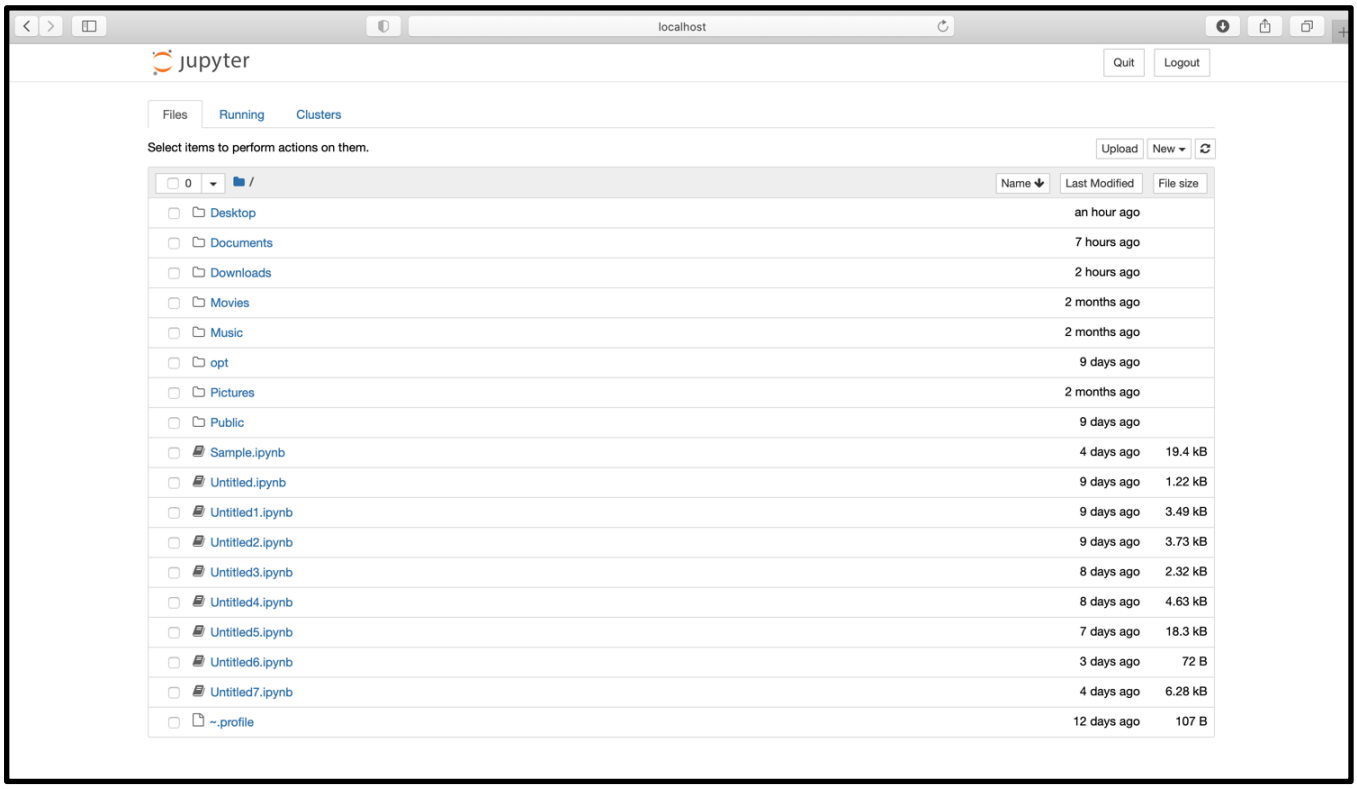

Image 2.6: Step 7

2.4. Pre-processing the Dataset

Dataset chosen for this study is not clean. In other words, it includes missing values, null values and duplications. Therefore, these inappropriate values are removed through pre-processing step.

- **Load the Data**

Load Dataset through DataFrame

The data set is loaded through the following steps:

- **Load the csv file into Dataframe with the code given below:**

dtFrame = spark.read.csv('/Users/manoj/desktop/employeeincomedetails.csv', header = True)

```
dtFrame = spark.read.csv('/Users/manoj/desktop/employeeincomedetails.csv', header = True)
```

Image 3.1: Upload the CSV file

- **Display the uploaded csv file by using below code.**

dtFrame.show (5)

```
dtFrame.show (5)
+---+---------+------+------------+-------------+--------------+-----------------+------------+-----+------+------
------+------------+--------------+--------------+------+
|age|workclass|fnlwgt|   education|education.num|marital.status|       occupation| relationship| race|   sex|capita
l.gain|capital.loss|hours.per.week|native.country|income|
+---+---------+------+------------+-------------+--------------+-----------------+------------+-----+------+------
------+------------+--------------+--------------+------+
| 90|        ?| 77053|     HS-grad|            9|       Widowed|                ?|Not-in-family|White|Female|
0|        4356|            40| United-States| <=50K|
| 82|  Private|132870|     HS-grad|            9|       Widowed|  Exec-managerial|Not-in-family|White|Female|
0|        4356|            18| United-States| <=50K|
| 66|        ?|186061|Some-college|           10|       Widowed|                ?|    Unmarried|Black|Female|
0|        4356|            40| United-States| <=50K|
| 54|  Private|140359|     7th-8th|            4|      Divorced|Machine-op-inspct|    Unmarried|White|Female|
0|        3900|            40| United-States| <=50K|
| 41|  Private|264663|Some-college|           10|     Separated|   Prof-specialty|    Own-child|White|Female|
0|        3900|            40| United-States| <=50K|
+---+---------+------+------------+-------------+--------------+-----------------+------------+-----+------+------
------+------------+--------------+--------------+------+
only showing top 5 rows
```

Image 3.2: Display the CSV file

- **To change the column name and display it from the uploaded file by using the below code.**

dtFrame2 = dtFrame.withColumnRenamed("age", "Age").withColumnRenamed("workclass", "Sector").withColumnRenamed("fnlwgt", "UID").withColumnRenamed("education", "Educational_Qualification").withColumnRenamed("education.num", "Education Grade").withColumnRenamed("marital.status", "Marital_Status").withColumnRenamed("occupation", "Occupation").withColumnRenamed("relationship", "Relationship").withColumnRenamed("race", "Race").withColumnRenamed("sex", "Gender").withColumnRenamed("capital.gain", "Capital Gain").withColumnRenamed("capital.loss", "Capital Loss").withColumnRenamed("native.country", "Country").withColumnRenamed('hours.per.week', 'Hours Per Week').withColumnRenamed("income", "Income")

```
dtFrame2 = dtFrame.withColumnRenamed("age", "Age").withColumnRenamed("workclass", "Sector").withColumnRenamed("fnlwg

dtFrame2.show(5)

+---+-------+------+------------------------+--------------+--------------+-----------------+------------+-----+
------+------------+------------+--------------+-------------+------+
|Age| Sector|   UID|Educational_Qualification|Education Grade|Marital_Status|       Occupation| Relationship| Race|
Gender|Capital Gain|Capital Loss|Hours Per Week|      Country|Income|
+---+-------+------+------------------------+--------------+--------------+-----------------+------------+-----+
------+------------+------------+--------------+-------------+------+
| 90|      ?| 77053|                 HS-grad|             9|       Widowed|                ?|Not-in-family|White|
Female|           0|        4356|            40|United-States| <=50K|
| 82|Private|132870|                 HS-grad|             9|       Widowed|  Exec-managerial|Not-in-family|White|
Female|           0|        4356|            18|United-States| <=50K|
| 66|      ?|186061|            Some-college|            10|       Widowed|                ?|    Unmarried|Black|
Female|           0|        4356|            40|United-States| <=50K|
| 54|Private|140359|                 7th-8th|             4|      Divorced|Machine-op-inspct|    Unmarried|White|
Female|           0|        3900|            40|United-States| <=50K|
| 41|Private|264663|            Some-college|            10|     Separated|   Prof-specialty|    Own-child|White|
Female|           0|        3900|            40|United-States| <=50K|
+---+-------+------+------------------------+--------------+--------------+-----------------+------------+-----+
------+------------+------------+--------------+-------------+------+
only showing top 5 rows
```

Image 3.3: Change the column name

- **Count the number of records.**

To count the number of records in the dataset by using the code shown below

dtFrame2.count()

```
dtFrame2.count()
32561
```

Image 3.4: Display the no. of Rows in the dataset

- **Dropping the Columns:**

After identifying all the counts in each column in the data by using the above mentioned code. We can drop the column which is not necessary for our analysis. From the updated data I have dropped some of the columns which is not useful for this analysis work. The dropped columns are **"Capital Gain, Capital Loss, UID, Hours Per Week, Education Grade"** and the code is mentioned below and the updated data is displayed as,

dtFrame2 = dtFrame2.drop('Capital Gain', 'Capital Loss', 'UID', 'Hours Per Week', 'Education Grade')

```
dtFrame2 = dtFrame2.drop('Capital Gain', 'Capital Loss', 'UID', 'Hours Per Week', 'Education Grade')

dtFrame2.show(5)
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+-------------+--
----+
|Age| Sector|Educational_Qualification|Marital_Status|       Occupation| Relationship| Race|Gender|      Country|In
come|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+-------------+--
----+
| 90|      ?|                 HS-grad|       Widowed|                ?|Not-in-family|White|Female|United-States| <
=50K|
| 82|Private|                 HS-grad|       Widowed|  Exec-managerial|Not-in-family|White|Female|United-States| <
=50K|
| 66|      ?|            Some-college|       Widowed|                ?|    Unmarried|Black|Female|United-States| <
=50K|
| 54|Private|                 7th-8th|      Divorced|Machine-op-inspct|    Unmarried|White|Female|United-States| <
=50K|
| 41|Private|            Some-college|     Separated|   Prof-specialty|    Own-child|White|Female|United-States| <
=50K|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+-------------+--
----+
only showing top 5 rows
```

Image 3.5: Removing a column from the dataset

- **Replacing the values in Machine readable form:**

The value in Income column in the defined data is mentioned as **"<=50K"** and **">50K"** in-order to understand it I have assigned those values as **"<=50K as 0"** and **">50K as 1"**. I have mentioned the code below.

dtFrame3 = dtFrame2.replace(['<=50K', '>50K'], ['0', '1'], 'Income')

```
dtFrame3 = dtFrame2.replace(['<=50K', '>50K'], ['0', '1'], 'Income')

dtFrame3.show(5)

+---+-------+------------------------+--------------+-----------------+-------------+-----+------+--------------+--
----+
|Age| Sector|Educational_Qualification|Marital_Status|       Occupation| Relationship| Race|Gender|       Country|In
come|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+--------------+--
----+
| 90|      ?|                 HS-grad|       Widowed|                ?|Not-in-family|White|Female|United-States|
0|
| 82|Private|                 HS-grad|       Widowed|  Exec-managerial|Not-in-family|White|Female|United-States|
0|
| 66|      ?|            Some-college|       Widowed|                ?|    Unmarried|Black|Female|United-States|
0|
| 54|Private|                 7th-8th|      Divorced|Machine-op-inspct|    Unmarried|White|Female|United-States|
0|
| 41|Private|            Some-college|     Separated|   Prof-specialty|    Own-child|White|Female|United-States|
0|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+--------------+--
----+
only showing top 5 rows
```

Image 3.6: Replace a value in the entire dataset

Also, replacing the **"?"** value in the entire data with **"Null"** value.

```
dtFrame3 = dtFrame3.replace('?', 'Null')

dtFrame3.show(5)

+---+-------+------------------------+--------------+-----------------+-------------+-----+------+--------------+--
----+
|Age| Sector|Educational_Qualification|Marital_Status|       Occupation| Relationship| Race|Gender|       Country|In
come|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+--------------+--
----+
| 90|   Null|                 HS-grad|       Widowed|             Null|Not-in-family|White|Female|United-States|
0|
| 82|Private|                 HS-grad|       Widowed|  Exec-managerial|Not-in-family|White|Female|United-States|
0|
| 66|   Null|            Some-college|       Widowed|             Null|    Unmarried|Black|Female|United-States|
0|
| 54|Private|                 7th-8th|      Divorced|Machine-op-inspct|    Unmarried|White|Female|United-States|
0|
| 41|Private|            Some-college|     Separated|   Prof-specialty|    Own-child|White|Female|United-States|
0|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+--------------+--
----+
only showing top 5 rows
```

- **Replacing the DataTypes:**

  Replacing the default datatypes for all the attributes to the preferred datatypes using the code mentioned below and displayed the datatypes for each column.

dtFrame4 = dtFrame3.withColumn("Age",
dtFrame3["Age"].cast(IntegerType())) .withColumn("Sector",

dtFrame3["Sector"].cast(StringType()))
.withColumn("Educational_Qualification",
dtFrame3["Educational_Qualification"].cast(StringType()))
.withColumn("Marital_Status",
dtFrame3["Marital_Status"].cast(StringType())) .withColumn("Occupation",
dtFrame3["Occupation"].cast(StringType())) .withColumn("Relationship",
dtFrame3["Relationship"].cast(StringType())) .withColumn("Gender",
dtFrame3["Gender"].cast(StringType())) .withColumn("Country",
dtFrame3["Country"].cast(StringType())) .withColumn("Income",
dtFrame3["Income"].cast(IntegerType())) .withColumn("Race",
dtFrame3["Race"].cast(StringType()))

```
dtFrame3
```

```
DataFrame[Age: string, Sector: string, Educational_Qualification: string, Marital_Status: string, Occupation: strin
g, Relationship: string, Race: string, Gender: string, Country: string, Income: string]
```

```
dtFrame4 = dtFrame3.withColumn("Age", dtFrame3["Age"].cast(IntegerType())) .withColumn("Sector", dtFrame3["Sector"].
```

```
dtFrame4.printSchema()
```

```
root
 |-- Age: integer (nullable = true)
 |-- Sector: string (nullable = true)
 |-- Educational_Qualification: string (nullable = true)
 |-- Marital_Status: string (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Relationship: string (nullable = true)
 |-- Race: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Income: integer (nullable = true)
```

Image 3.7: Replacing the Datatypes in the dataset

- **Removing the Unwanted Row:**

Removing the “Null” present in few cell and deleting the entire row by using the code mentioned below.

dtFrame5 = dtFrame4.filter((dtFrame4.Sector != 'Null') & (dtFrame4.Occupation != 'Null') & (dtFrame4.Relationship != 'Null') & (dtFrame4.Gender != 'Null') & (dtFrame4.Country != 'Null') & (dtFrame4.Educational_Qualification != 'Null') & (dtFrame4.Marital_Status != 'Null'))

```
dtFrame4.select('Age', 'Sector', 'Educational_Qualification', 'Marital_Status', 'Occupation', 'Relationship', 'Gende
32561

(dtFrame4.Country != 'Null') & (dtFrame4.Educational_Qualification != 'Null') & (dtFrame4.Marital_Status != 'Null'))

dtFrame5.show(5)

+---+-------+------------------------+--------------+-----------------+-------------+-----+------+-------------+--
----+
|Age| Sector|Educational_Qualification|Marital_Status|       Occupation| Relationship| Race|Gender|      Country|In
come|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+-------------+--
----+
| 82|Private|                 HS-grad|       Widowed|  Exec-managerial|Not-in-family|White|Female|United-States|
0|
| 54|Private|                 7th-8th|      Divorced|Machine-op-inspct|    Unmarried|White|Female|United-States|
0|
| 41|Private|            Some-college|     Separated|   Prof-specialty|    Own-child|White|Female|United-States|
0|
| 34|Private|                 HS-grad|      Divorced|    Other-service|    Unmarried|White|Female|United-States|
0|
| 38|Private|                    10th|     Separated|     Adm-clerical|    Unmarried|White|  Male|United-States|
0|
+---+-------+------------------------+--------------+-----------------+-------------+-----+------+-------------+--
----+
only showing top 5 rows


dtFrame5.select('Age', 'Sector', 'Educational_Qualification', 'Marital_Status', 'Occupation', 'Relationship', 'Gende
30162
```

Image 3.8: Deleting the Rows in the dataset

- **Grouping the Data Values:**

Grouping the all the column values in the dataset and arranging them in the descending order using the code mentioned below.

dtFrame5.groupBy('Age', 'Sector', 'Educational_Qualification', 'Marital_Status', 'Relationship', 'Gender', 'Country', 'Income', 'Race').count().sort(desc("count")).show(10)

```
dtFrame5.groupBy('Age', 'Sector', 'Educational_Qualification', 'Marital_Status', 'Relationship', 'Gender', 'Country'

+---+-------+------------------------+------------------+------------+------+-------------+------+-----+-----+
|Age| Sector|Educational_Qualification|    Marital_Status|Relationship|Gender|      Country|Income| Race|count|
+---+-------+------------------------+------------------+------------+------+-------------+------+-----+-----+
| 20|Private|            Some-college|     Never-married|   Own-child|Female|United-States|     0|White|   98|
| 19|Private|            Some-college|     Never-married|   Own-child|Female|United-States|     0|White|   95|
| 20|Private|            Some-college|     Never-married|   Own-child|  Male|United-States|     0|White|   85|
| 21|Private|            Some-college|     Never-married|   Own-child|Female|United-States|     0|White|   77|
| 19|Private|                 HS-grad|     Never-married|   Own-child|  Male|United-States|     0|White|   74|
| 21|Private|            Some-college|     Never-married|   Own-child|  Male|United-States|     0|White|   72|
| 35|Private|                 HS-grad|Married-civ-spouse|     Husband|  Male|United-States|     0|White|   70|
| 33|Private|                 HS-grad|Married-civ-spouse|     Husband|  Male|United-States|     0|White|   64|
| 19|Private|            Some-college|     Never-married|   Own-child|  Male|United-States|     0|White|   64|
| 36|Private|                 HS-grad|Married-civ-spouse|     Husband|  Male|United-States|     0|White|   63|
+---+-------+------------------------+------------------+------------+------+-------------+------+-----+-----+
only showing top 10 rows
```

Image 3.9: Grouping the Data Values

- **Downloading the Processed Dataset:**

After removing all the unwanted data and changing the values the **"Cleaned Data"** can be saved into the system as **"CSV file"** using the below code.

dtFrame5.write.csv('/Users/manoj/desktop/Course Work -- BD & DV.csv',header=True)

```
dtFrame5.write.csv('/Users/manoj/desktop/Course Work -- BD & DV.csv',header=True)
```

Image 3.10: Downloading the processed dataset

This **.csv file** is used for data visualisation in **Tableau** and it is further discussed and analysed below.

2.5. Data Visualisation

The processed dataset is used which is obtained from the Jupyter Notebook is visualised using Tableau. Let us discuss in detail about the installation procedure of Tableau, how the data is visualised in Tableau.

Installation Steps (Tabulae):

Tableau is an interactive data visualization software. It will help to generate graph-type data visualizations. Download the Tableau Desktop 2020.3 from their website and install it in the Desktop/laptop which is used to generate the visualisation. After installing the Tableau upload the processed dataset in to it. Then select "Use Data Interpreter" checkbox to remove the "Null" rows in the processed dataset. After this process the data can be used for visualisation.

Countries Geographical Representation:

Location Analysis is a study about different economic factors associated to locations in a country. In this course work, individual employee's income status of different countries are chosen to find the relationship among the factors and to predict the employee's income status of each country.

The study has consider specific country and that is represented in the map below with the aid of actual “Longitude and Latitude” also the legends indicates the specific country with colour assigned to it.

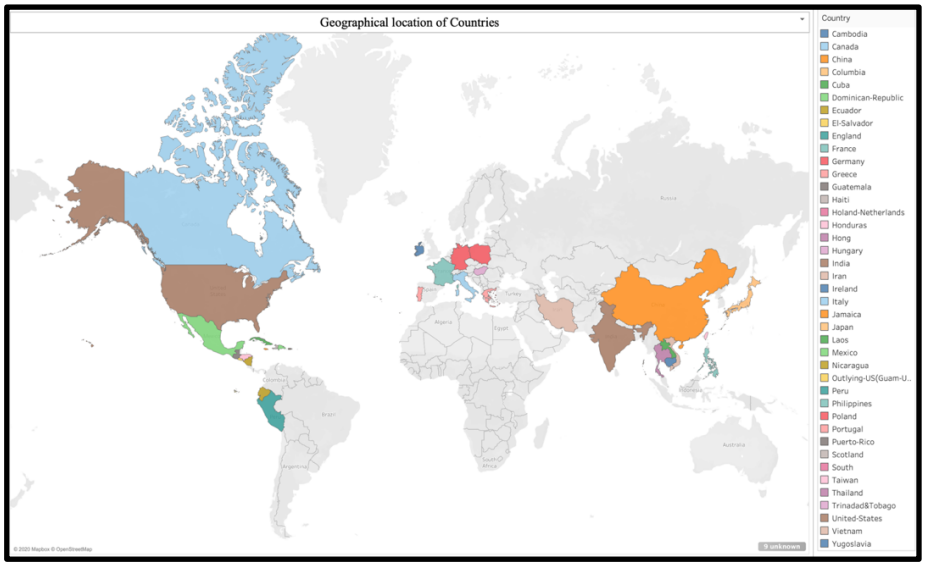

Image 4.1: Country Chart

Location vs. Gender:

Based on this study, the below chart explains link between the countries and total number of Male and the Female who were employed in the particular countries. In this percentage of male or female were represented in blue and orange “Dots” respectively.

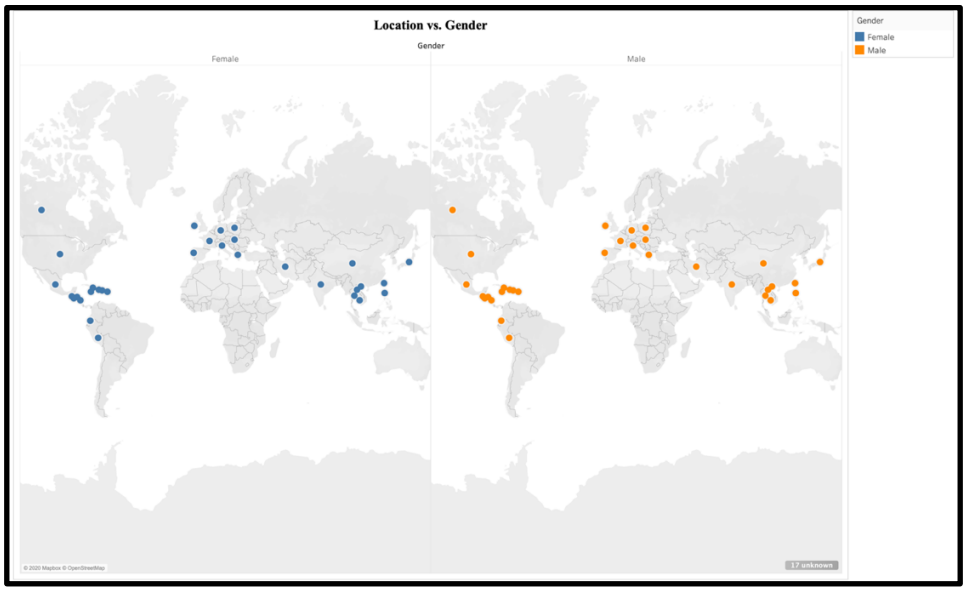

Image 4.2: Location vs. Gender

Educational Qualification vs. Income Count:

Next, the link between Educational Qualification and Income count attributes were explored and it is illustrated in the "Bar chart" shown below.

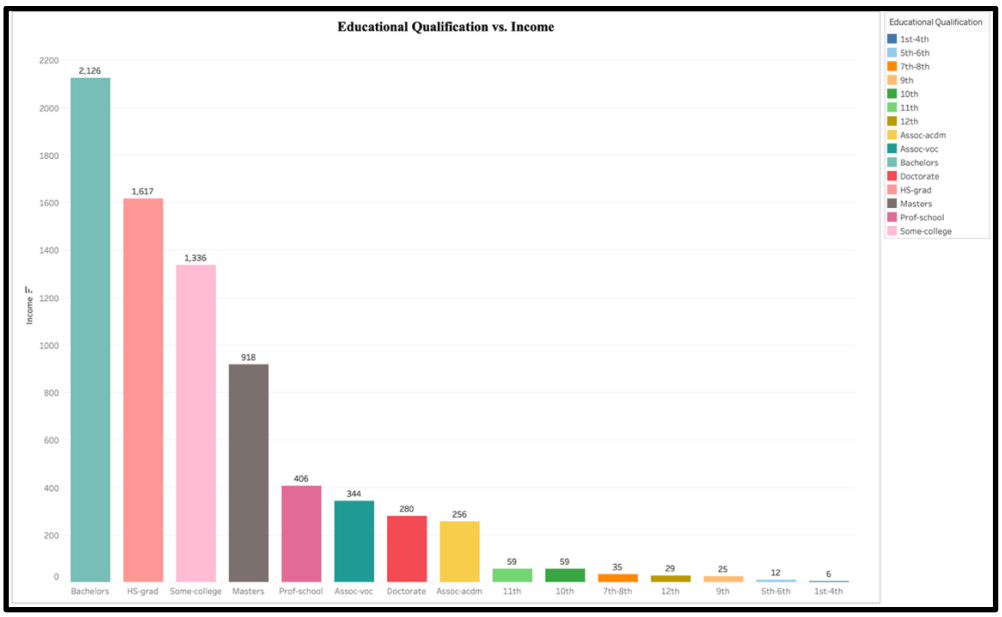

Image 4.3: Educational Qualification vs. Income Count

Marital Status vs. Income Count:

Next, the link between Marital Status of the people and their Income count were explored and it is demonstrated in the "Bar chart" shown below.

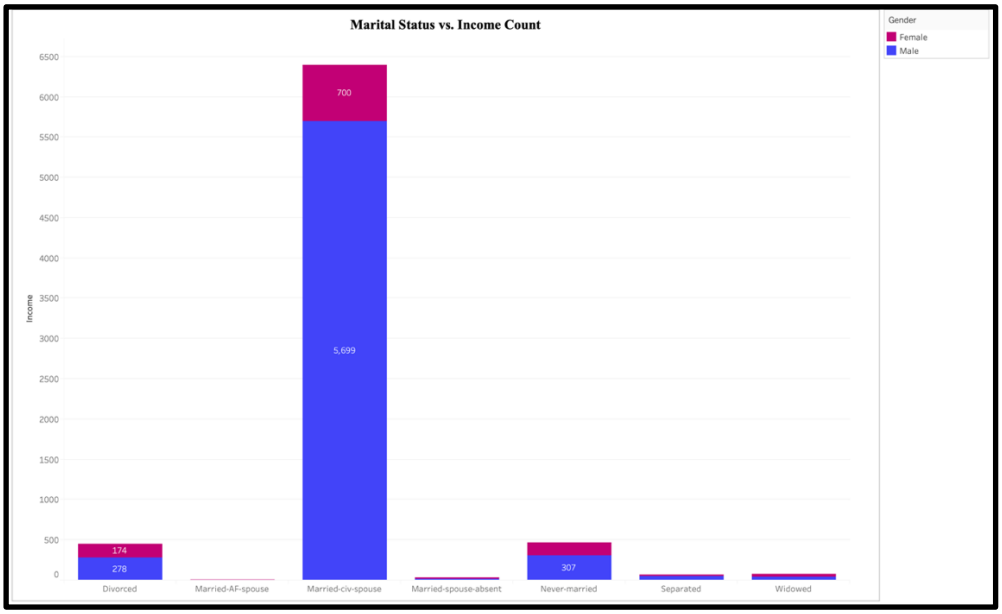

Image 4.4: Marital Status vs. Income Count

Occupation vs. Age vs. Income Count:

Following this, the link between Occupation of the people and total Age count of the people were explored and it is demonstrated in the “Bar chart” and the “Line chart” which demonstrate the link between Occupation of the people and total Age count of the people along with their Income count.

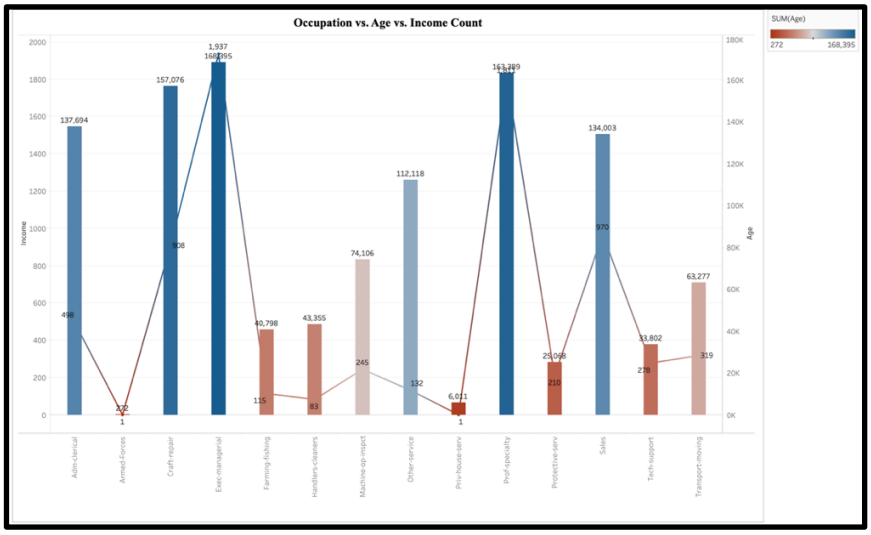

Image 4.5: Occupation vs. Age vs. Income Count

Race vs. Work Sector vs. Income Count:

Following this, the link between Race based on region of the people and the working sector of those people and their Income Count were explored and it is demonstrated in the “Bar chart”.

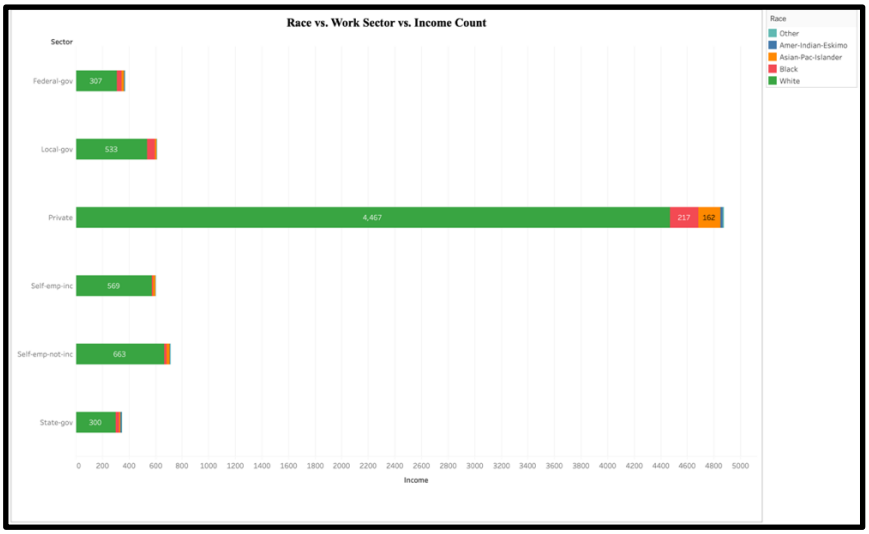

Image 4.6: Race vs. Work Sector vs. Income Count

Work Sector vs. Gender vs. Income Count:

The below describes about the connections between three attributes which are Sector, Gender and Income. This is shown in the following "Bar chart" with the comparison between the Male and Female gender.

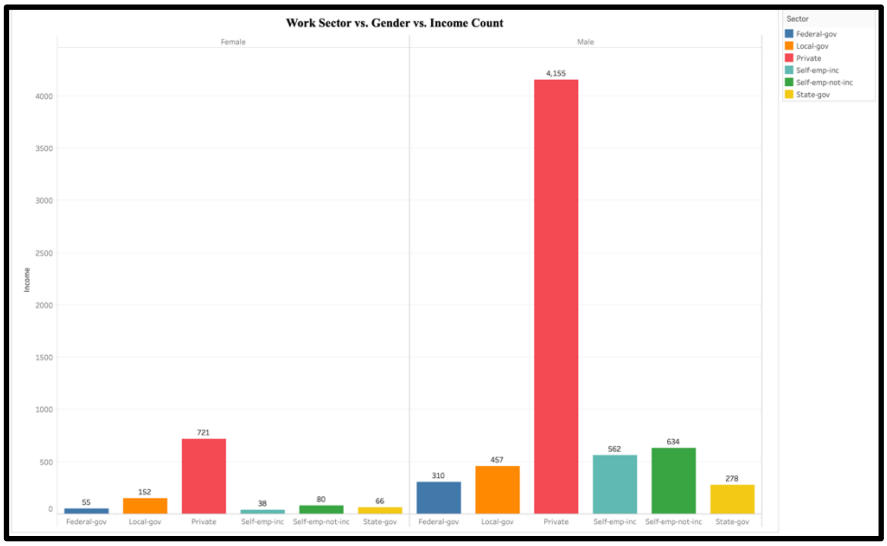

Image 4.7: Work Sector vs. Gender vs. Income Count

Race vs. Occupation vs. Count of Educational Qualification:

The below chart describes about the connections between Race based on region of the people, Occupation and Educational Qualification. This is shown in the following "Shape (circle) chart" with different colours illustrates the Race.

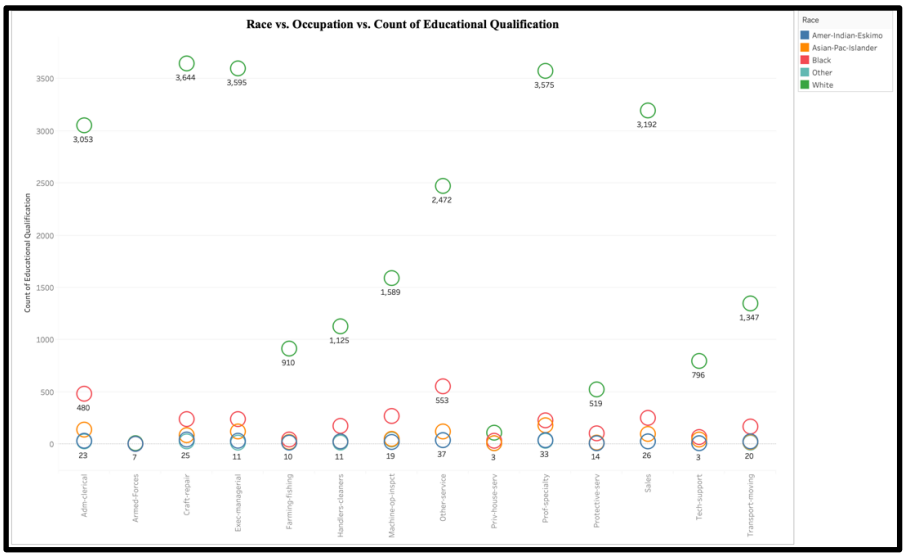

Image 4.8: Race vs. Occupation vs. Count of Educational Qualification

Occupation vs. Sector vs. Gender vs. Income:

Then this analysis has moved further advance through looking at the connections between four attributes which are Occupation, Sector, Gender and Income. This is shown in the following different "Shape chart", where the X-axis represents the Count of the Gender and Y-axis represents the Income.

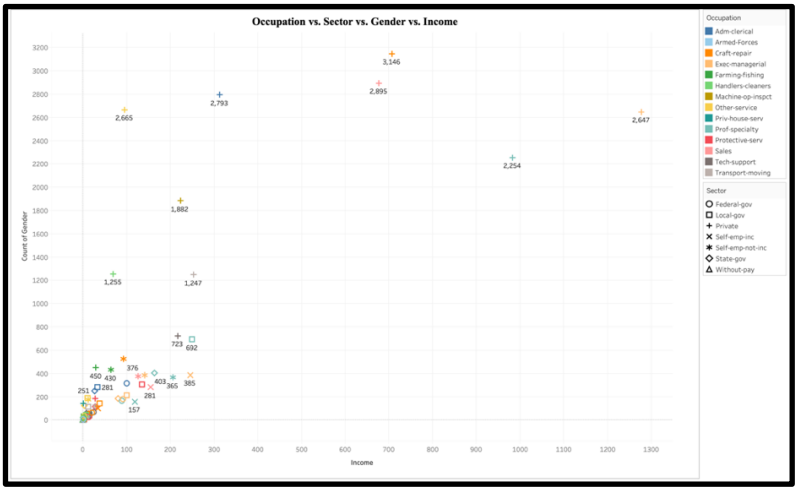

Image 4.9: Occupation vs. Sector vs. Gender vs. Income

Country vs. Educational Qualification vs. Gender vs. Work Sector vs. Occupation:

Again, this analysis has moved further advance through looking at the connections between five attributes which are Country, Educational Qualification, Gender, Work Sector and Occupation . This is shown in the following using "Bubble chart".

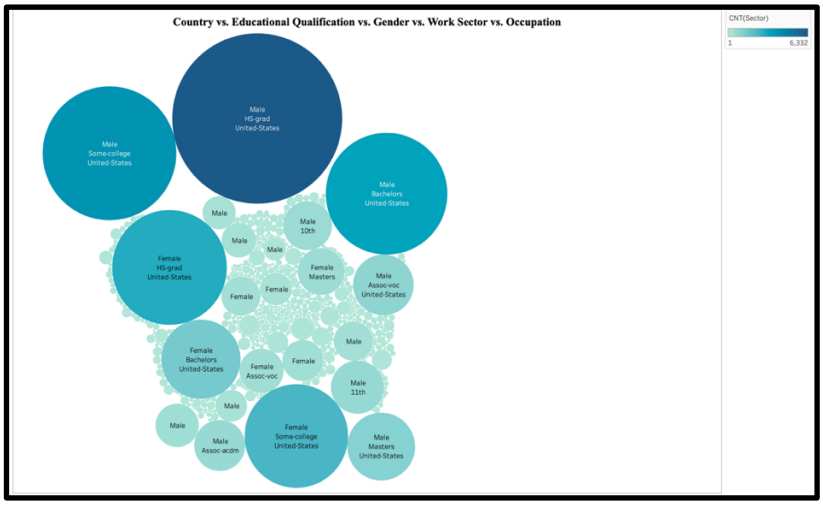

Image 4.10: Country vs. Educational Qualification vs. Gender vs. Work Sector vs. Occupation

2.6. Machine Learning – Linear Regression

Machine learning is the process by which a computer can work more accurately as it collects and learns from the data it is given. Linear Regression is a type of supervised machine learning algorithm for predicting continuous-valued output where the predicted output is continuous and has a constant slope. It's used to predict values more effectively and efficiently. In this study we have generated prediction value and correlation matrix to identify the Linear Regression between "Age" and "Income" which are Integer datatypes. The dataset has been split into 70% training dataset and 30% testing dataset and the transformation through vector assembler to setup "Income", "Age" as features and labels. This is explained in the below code.

The following code has been used for it.

- Importing the cleaned dataset to identify the prediction value and correlation matrix. Here the cleaned dataset has been read using spark and saved in the Data Frame "dtFrame". Using the below mentioned code the rows and the column of the dataset has been displayed.

print((dtFrame.count(), len(dtFrame.columns)))

```
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.functions import corr
spark = SparkSession.builder.appName('lrex').getOrCreate()
```

```
dtFrame = spark.read.csv(path='/Users/manoj/desktop/cleaneddata.csv', header = True, inferSchema = True)
```

```
print((dtFrame.count(), len(dtFrame.columns)))
```

```
(30162, 10)
```

```
dtFrame.printSchema()
```

```
root
 |-- Age: integer (nullable = true)
 |-- Sector: string (nullable = true)
 |-- Educational_Qualification: string (nullable = true)
 |-- Marital_Status: string (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Relationship: string (nullable = true)
 |-- Race: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Income: integer (nullable = true)
```

Image 5.1: Loading the processed dataset

- To identify and to calculate the summary statistics of numerical column from the dataset the below mention code is used.

dfFrame1 = dtFrame.describe().show(5)

```
dfFrame1 = dtFrame.describe().show(5)
+-------+------------------+-----------+------------------------+--------------+----------------+------------+----
-------------+------+----------+-------------------+
|summary|               Age|     Sector|Educational_Qualification|Marital_Status|      Occupation|Relationship|
Race|Gender|   Country|             Income|
+-------+------------------+-----------+------------------------+--------------+----------------+------------+----
-------------+------+----------+-------------------+
|  count|             30162|      30162|                   30162|         30162|           30162|       30162|
30162| 30162|     30162|              30162|
|   mean|38.437901995888865|       null|                    null|          null|            null|        null|
null|  null|      null|0.24892248524633645|
| stddev|13.134664776856031|       null|                    null|          null|            null|        null|
null|  null|      null| 0.4323959763009978|
|    min|                17|Federal-gov|                    10th|      Divorced|    Adm-clerical|     Husband|Amer
-Indian-Eskimo|Female|  Cambodia|                  0|
|    max|                90|Without-pay|            Some-college|       Widowed|Transport-moving|        Wife|
White|  Male|Yugoslavia|                  1|
+-------+------------------+-----------+------------------------+--------------+----------------+------------+----
-------------+------+----------+-------------------+
```

Image 5.2: Displaying the processed dataset

- To identify the correlation between the two numeric columns in the dataset by using the below mentioned code.

dtFrame1.select(corr('Age','Income')).show()

```
dtFrame1.select(corr('Age','Income')).show()
+--------------------+
|  corr(Age, Income)|
+--------------------+
|0.24199813626611658|
+--------------------+
```

Image 5.3: Correlation Value

- Assigning the input value and output value in assembler using VectorAssembler feature. Then assigning the random split value as (0.7 and 0.3) for features and Age respectively. The code for this is mentioned below.

assembler = VectorAssembler(inputCols=['Income'], outputCol='features')

dtFrame2 = assembler.transform(dtFrame)

dtFrame2 = dtFrame2.select(['features','Age'])

train,test = dtFrame2.randomSplit([0.7, 0.3])

train.show(5)

test.show(5)

```
assembler = VectorAssembler(inputCols=['Income'], outputCol='features')
dtFrame2 = assembler.transform(dtFrame)
```

```
dtFrame2 = dtFrame2.select(['features','Age'])
```

```
train,test = dtFrame2.randomSplit([0.7, 0.3])
```

```
train.show(5)
```

```
+--------+---+
|features|Age|
+--------+---+
|   [0.0]| 17|
|   [0.0]| 17|
|   [0.0]| 17|
|   [0.0]| 17|
|   [0.0]| 17|
+--------+---+
only showing top 5 rows
```

```
test.show(5)
```

```
+--------+---+
|features|Age|
+--------+---+
|   [0.0]| 17|
|   [0.0]| 17|
|   [0.0]| 17|
|   [0.0]| 17|
|   [0.0]| 17|
+--------+---+
only showing top 5 rows
```

Image 5.4: Assigning the Input and Output values

- Calculating Intercept and Coefficient using the assigned LinearRegression value.

```
lr = LinearRegression(featuresCol = 'features', labelCol='Age')
lrModel = lr.fit(train)
```

```
print(f'Intercept: {lrModel.intercept}\nCoefficient: {lrModel.coefficients.values}')
```

```
Intercept: 36.690519456106266
Coefficient: [7.23600961]
```

Image 5.5: Intercept and Coefficient values

- Calculating the prediction and accuracy values using the assigned feature and label column using the below mentioned code.

```
from  pyspark.sql.functions import abs
predictions = lrModel.transform(test)
x =((predictions['Age']-predictions['prediction'])/predictions['Age'])*100
predictions = predictions.withColumn('Accuracy',abs(x))
predictions.select("prediction","Age","Accuracy","features").show(5)
```

```
+------------------+---+------------------+--------+
|        prediction|Age|          Accuracy|features|
+------------------+---+------------------+--------+
|36.690519456106266| 17|115.82658503591921|   [0.0]|
|36.690519456106266| 17|115.82658503591921|   [0.0]|
|36.690519456106266| 17|115.82658503591921|   [0.0]|
|36.690519456106266| 17|115.82658503591921|   [0.0]|
|36.690519456106266| 17|115.82658503591921|   [0.0]|
+------------------+---+------------------+--------+
only showing top 5 rows
```

Image 5.6: Prediction and Accuracy values

- After then the predicted values were evaluated through R^2 values and achieved approximately 0.06%. Higher the R^2 better the model.

```
from pyspark.ml.evaluation import RegressionEvaluator
pred_evaluator = RegressionEvaluator(predictionCol="prediction", labelCol="Age",metricName="r2")
print("R Squared (R2) on test data = %g" % pred_evaluator.evaluate(predictions))
```

```
R Squared (R2) on test data = 0.0630872
```

```
r2 = trainSummary.r2
n = dtFrame.count()
p = len(dtFrame.columns)
adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
```

```
lin_reg = LinearRegression(featuresCol = 'features', labelCol='Age',maxIter=50, regParam=0.12, elasticNetParam=0.2)
linear_model = lin_reg.fit(train)
```

```
linear_model.summary.rootMeanSquaredError
```

```
12.77078802265797
```

Image 5.7: R2 value

2.7. Discussion:

The following discussion for this study is based on the data analytics, data visualisation and machine learning algorithms. I have demonstrated positive linear relationship between the Age and Income. Moreover, this is also confirmed through the values of intercepts and co-efficient which are 36.6095 and 7.236 respectively. Generally, when the R2 is high the linear regression model will perform well. However, in our study the value of R2 is 0.06 hence, it is a normal performing linear regression model.

In the above mentioned "Location vs. Gender" data visualisation chart, it explains about the relationship between the countries and the employees in the particular region. Here, total number for both male and female are high in the Americas region (North America and South America). After focusing on the location, "Educational Qualification vs. Income" chart demonstrate about the employees who had completed Bachelor's degree are paid higher and on the other hand, the low paid employees had a qualification of standard 1^{st} - 4^{th}.

Education plays a major role in the county's growth, economy and development. In this study, it was examined through the chart "Country vs. Educational Qualification vs. Gender vs. Work Sector vs. Occupation" that most of employees who had completed their graduate degree are employed in most of the sectors and are paid in higher hands. Based on the analysis, it is evident that United States of Americas has more number of educated employees.

Another finding from this study was also visualized in the chart "Work Sector vs. Gender vs. Income Count" as this demonstrates the comparison between both the Male and Female employees who were working in the different sectors. The comparison chart shows that male employees who were working in all sectors were higher than the female employees who working in the same sectors. In particular, private sector secured the higher number for both male and female employees.

2.8. Conclusion:

The results from the visualisations and discussion shows overall there is a positive relationship between Age and Income. Americas ranks the top region, where the number of employees who are employed in all the sector and also the Male and Female who contributed much also falls under the same region. On the other hand, Asia Pacific reached the lowest level in all the sector. This shows that employee in the American region are wealthy enough with the good educational background. This illustrate that these factors determine the wealth of the region.

Overall, this study concludes that each factors has its own impact on economic development hence it is much vital to focus on all the factors individually.

2.9. References:

Guy Harrison (2015), "*Next Generation Databases*" [Online]

Flannery O'Connor, "*The Internals of Spark SQL (Apache Spark 2.4.5)*" [Online]

Swatee Chand (2020), "*PySpark Programming – Integrating Speed With Simplicity*" [Online]

Chris A. Mattmann (2020), "*Machine Learning with TensorFlow, Second Edition*" [Online]

Davy Cielen, Arno Meysman, Mohamed Ali (2016), "*Introducing Data Science - Big Data, Machine Learning, and More, Using Python Tools*" [Online]

Adult income dataset. kaggle. [Online] Available from: <https://www.kaggle.com/wenruliu/adult-income-dataset> [06.10. 2016].

Java 1.8.0_261 [Online] < https://www.java.com/en/download/>

Spark version 3.0.1 [Online] <https://spark.apache.org/downloads.html>

Anaconda Installers (Python 3.8) [Online] <https://www.anaconda.com/products/individual#macos>

*Tableau Desktop 2020.*3 [Online] < https://www.tableau.com/en-gb/products/desktop/download>

PySpark Flow Diagram [Online] < https://www.researchgate.net/figure/An-illustration-of-PySpark-data-flow-The-Python-environments-were-shaded-in-white-and_fig2_325055629>

Somesh Routray (2020), "*Machine Learning : Linear Regression using Pyspark*" [Online] < https://towardsdatascience.com/machine-learning-linear-regression-using-pyspark-9d5d5c772b42>

3. Appendix

Please see the link below to access the codes used for this study.
<https://github.com/Manojsibi/Manoj_SM-Projects.git>