Term Project
Lambton College - AML 1114_1
Exploratory data analysis on the adult dataset

Team Members

Bhanu Prakash Mahadevuni – C0850515
Deeksha Naikap – C0835440
Pramod Reddy Gurrala – C0850493
Sai Varun Kollipara – C0828403

Problem Statement

The problem states to apply exploratory analysis to one of the UCI machine learning datasets: the heart disease dataset or adult dataset. This includes performing the analysis using Python, finding out missing values, and handling them. Finally, based on the observation, decide how to use the dataset for further analysis: regression or classification.

Solution Approach

The approach for this case is to consider the adult dataset from Kaggle and apply exploratory analysis to it. Also, to include finding missing values and handling them using different techniques.

Steps Involved

These are the steps involved in the project study:

1. Importing the Packages and Libraries, and Functions
2. Importing and Loading the Dataset
3. Exploratory Analysis
4. Data Pre-processing
5. Data Visualization
6. Feature Evaluation

System Requirements / Software Requirements

- Jupyter Notebooks
- pandas
- numpy
- matplotlib
- seaborn

Data Pre-processing

Data Pre-processing consists of multiple phases where data is cleaned, altered, and made suitable for the machine learning models to enhance the performance.

Here are some steps implemented:

- Performing the statistical analysis on the dataset
- Checking for missing values
- Handling missing values

Loading and viewing the dataset:

Dataset – Adult dataset

https://archive.ics.uci.edu/ml/datasets/adult

[3]:
```
income_df = pd.read_csv("./archive/adult.csv")
```

[4]:
```
income_df
```

[4]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

48842 rows × 15 columns

Understanding the dataset size and datatypes:

- The Adult dataset has 15 attributes (columns) and 48842 instances (records). of 3 multivalued discrete and 5 continuous attributes.
- It consists of both categorical and numerical attributes.
- Work class, education, marital-status, occupation, relationship, race, sex, and native-country are the categorical attributes.
- Age, Fnlwgt, capital-gain, capital-loss, and hours-per-week are the continuous numerical attributes.

```
[7]: income_df.shape

[7]: (48842, 15)

[8]: income_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              48842 non-null  int64 
 1   workclass        48842 non-null  object
 2   fnlwgt           48842 non-null  int64 
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64 
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64 
 11  capital-loss     48842 non-null  int64 
 12  hours-per-week   48842 non-null  int64 
 13  native-country   48842 non-null  object
 14  income           48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

Performing statistical analysis:

```
[10]: income_df.describe()
```

[10]:

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

Missing values

Missing values are defined as not available values, and that would be meaningful if observed. Missing values can be anything from missing sequence, incomplete feature, files missing, information incomplete, data entry error etc. Most datasets in the real world contain missing values.

Checking for Null values:

```
[11]: income_df.isnull().sum()

[11]: age                  0
      workclass            0
      fnlwgt               0
      education            0
      educational-num      0
      marital-status       0
      occupation           0
      relationship         0
      race                 0
      gender               0
      capital-gain         0
      capital-loss         0
      hours-per-week       0
      native-country       0
      income               0
      dtype: int64
```

```
[12]: income_df.isin(['?']).sum()

[12]: age                     0
      workclass            2799
      fnlwgt                  0
      education               0
      educational-num         0
      marital-status          0
      occupation           2809
      relationship            0
      race                    0
      gender                  0
      capital-gain            0
      capital-loss            0
      hours-per-week          0
      native-country        857
      income                  0
      dtype: int64
```

Since there is multiple '?' in the dataset, we created a new variable to save the dataset for testing purposes.

```
[13]: data_num = income_df.copy()
```

Handling missing values in data

Main approaches to deal with missing values-

Omission

Remove the columns and rows containing missing values or invalid data for further analysis. It creates a subset of the dataset with no missing values and works well for models that are not robust against data missingness.

Dropping the column:

```
[15]: data_num = data_num.drop(columns='native-country')
```

```
[16]: data_num.head()
```

[16]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	<=50K

Imputation

Missing data is replaced/filled with other values in the data set. This method works well for situations where analysis tools are not robust to missing values. Dataset sizes are not reduced but the noise gets imposed with the imputation

Imputation Techniques (mode function since string):

```
7]: attrib, counts = np.unique(data_num['workclass'], return_counts = True)
most_freq_attrib = attrib[np.argmax(counts, axis = 0)]
data_num['workclass'][data_num['workclass'] == '?'] = most_freq_attrib

attrib, counts = np.unique(data_num['occupation'], return_counts = True)
most_freq_attrib = attrib[np.argmax(counts, axis = 0)]
data_num['occupation'][data_num['occupation'] == '?'] = most_freq_attrib
```

Nan values were as ‘?’ in the data. Hence, we fix this with the corresponding most frequent element(mode) in the entire dataset. It generalizes well, as we will see with the accuracy.

```
]: data_num
```

]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	>50K
4	18	Private	103497	Some-college	10	Never-married	Prof-specialty	Own-child	White	Female	0	0	30	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	<=50K
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	>50K
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	<=50K
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	<=50K
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	>50K

48842 rows × 14 columns

Duplication Identification

```
9]:  print(f"We have {data_num.duplicated().sum()} duplicate values")
```

```
We have 58 duplicate values
```

Datasets containing duplicates may contaminate training data with the test data or vice versa, and effects the accuracy and performance of the machine learning model.

Drop duplicates

```
[20]:  data_num = data_num.drop_duplicates()

       print(f"After dropping duplicate values, now we have {data_num.duplicated().sum()} duplicate values")
```

```
After dropping duplicate values, now we have 0 duplicate values
```

Feature Engineering

```
[21]:  print('workclass',data_num.workclass.unique())
       print('education',data_num.education.unique())
       print('marital-status',data_num['marital-status'].unique())
       print('occupation',data_num.occupation.unique())
       print('relationship',data_num.relationship.unique())
       print('race',data_num.race.unique())
       print('gender',data_num.gender.unique())
       print('income',data_num.income.unique())
```

```
workclass ['Private' 'Local-gov' 'Self-emp-not-inc' 'Federal-gov' 'State-gov'
 'Self-emp-inc' 'Without-pay' 'Never-worked']
education ['11th' 'HS-grad' 'Assoc-acdm' 'Some-college' '10th' 'Prof-school'
 '7th-8th' 'Bachelors' 'Masters' 'Doctorate' '5th-6th' 'Assoc-voc' '9th'
 '12th' '1st-4th' 'Preschool']
marital-status ['Never-married' 'Married-civ-spouse' 'Widowed' 'Divorced' 'Separated'
 'Married-spouse-absent' 'Married-AF-spouse']
occupation ['Machine-op-inspct' 'Farming-fishing' 'Protective-serv' 'Prof-specialty'
 'Other-service' 'Craft-repair' 'Adm-clerical' 'Exec-managerial'
 'Tech-support' 'Sales' 'Priv-house-serv' 'Transport-moving'
 'Handlers-cleaners' 'Armed-Forces']
relationship ['Own-child' 'Husband' 'Not-in-family' 'Unmarried' 'Wife' 'Other-relative']
race ['Black' 'White' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']
gender ['Male' 'Female']
income ['<=50K' '>50K']
```

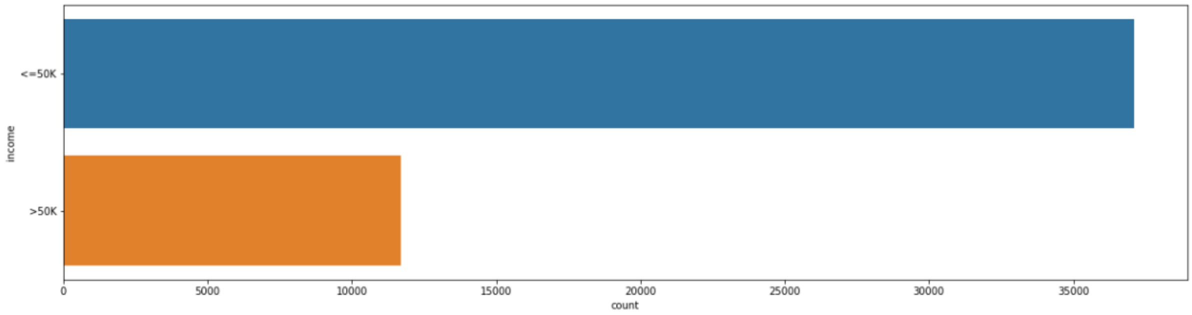
Data visualization

Data Visualization helps in understanding the data with pictorial representation. Some of the visualizations that can be performed using ML libraries are listed here:

- Bar- Graph
- Histogram
- Pie-chart
- Heat-map
- 3D-plotting and many more

Income

```
<AxesSubplot:xlabel='count', ylabel='income'>
```



This distribution says that the majority of them belong to the income group ‘<=50K’ (who earns less than 50k) and the rest fall under the other income group (who earns more than 50k).

Education

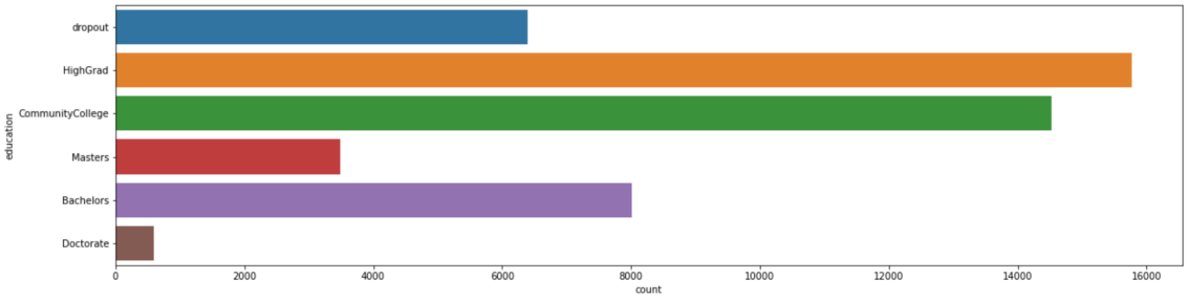
```
data_num['education'].replace('Preschool', 'dropout',inplace=True)
data_num['education'].replace('10th', 'dropout',inplace=True)
data_num['education'].replace('11th', 'dropout',inplace=True)
data_num['education'].replace('12th', 'dropout',inplace=True)
data_num['education'].replace('1st-4th', 'dropout',inplace=True)
data_num['education'].replace('5th-6th', 'dropout',inplace=True)
data_num['education'].replace('7th-8th', 'dropout',inplace=True)
data_num['education'].replace('9th', 'dropout',inplace=True)
data_num['education'].replace('HS-Grad', 'HighGrad',inplace=True)
data_num['education'].replace('HS-grad', 'HighGrad',inplace=True)
data_num['education'].replace('Some-college', 'CommunityCollege',inplace=True)
data_num['education'].replace('Assoc-acdm', 'CommunityCollege',inplace=True)
data_num['education'].replace('Assoc-voc', 'CommunityCollege',inplace=True)
data_num['education'].replace('Bachelors', 'Bachelors',inplace=True)
data_num['education'].replace('Masters', 'Masters',inplace=True)
data_num['education'].replace('Prof-school', 'Masters',inplace=True)
data_num['education'].replace('Doctorate', 'Doctorate',inplace=True)
```

```
data_num[['education', 'educational-num']].groupby(['education'], as_index=False).mean().sort_values(by='educational-num', ascending=False)
```

Out[24]:

	education	educational-num
2	Doctorate	16.000000
4	Masters	14.238968
0	Bachelors	13.000000
1	CommunityCollege	10.362372
3	HighGrad	9.000000
5	dropout	5.618512

```
<AxesSubplot:xlabel='count', ylabel='education'>
```

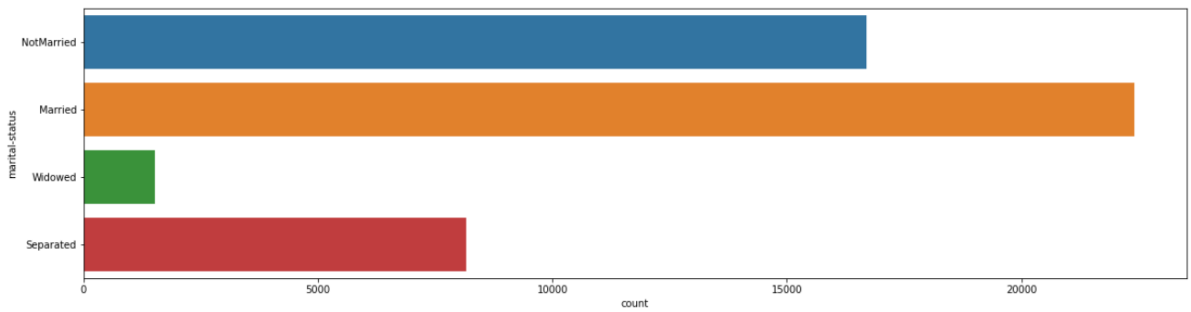


From the plot distribution, it is clear that the number of high school graduates is very high whereas, the number of doctorate holders is the least.

Marital Status

```
data_num['marital-status'].replace('Never-married', 'NotMarried',inplace=True)
data_num['marital-status'].replace(['Married-AF-spouse'], 'Married',inplace=True)
data_num['marital-status'].replace(['Married-civ-spouse'], 'Married',inplace=True)
data_num['marital-status'].replace(['Married-spouse-absent'], 'NotMarried',inplace=True)
data_num['marital-status'].replace(['Separated'], 'Separated',inplace=True)
data_num['marital-status'].replace(['Divorced'], 'Separated',inplace=True)
data_num['marital-status'].replace(['Widowed'], 'Widowed',inplace=True)
```

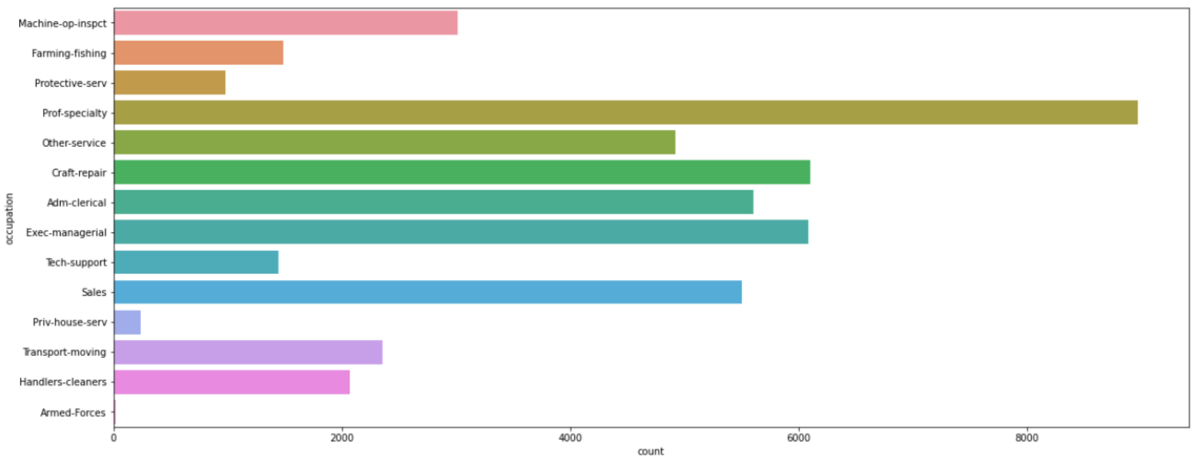
```
<AxesSubplot:xlabel='count', ylabel='marital-status'>
```



The ‘Not Married’ category dominates over other categories. Married has the maximum number of samples. And widowed has the minimum number of obs.

Occupation

```
<AxesSubplot:xlabel='count', ylabel='occupation'>
```



There are 14 unique categories present in the occupation attribute. Prof-specialty has the maximum count, but armed-Forces have minimum samples in the occupation attribute.

Age

Created age_bin to categorize the age and use for visualizing.

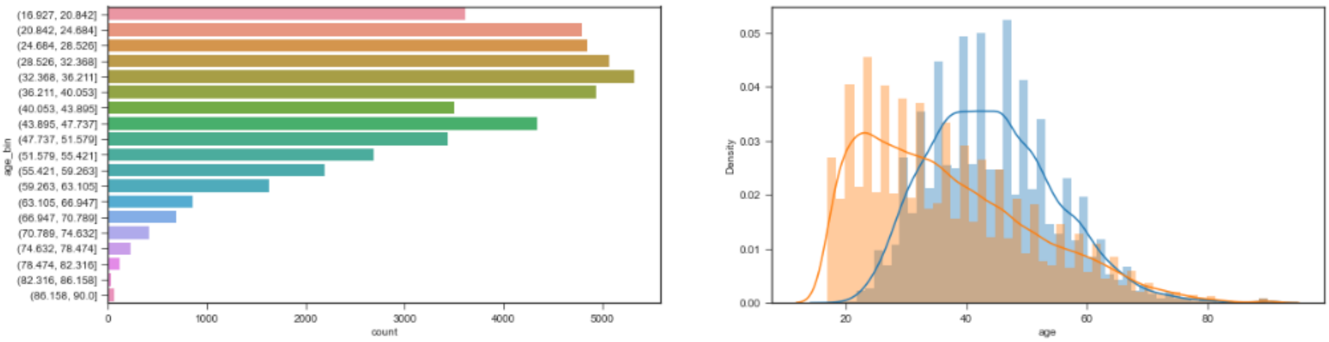
```
data_num['age_bin'] = pd.cut(data_num['age'], 19)
```

```
30]: data_num['age_bin']
```

```
30]: 0        (24.684, 28.526]
     1        (36.211, 40.053]
     2        (24.684, 28.526]
     3        (43.895, 47.737]
     4        (16.927, 20.842]
                    ...
     48837    (24.684, 28.526]
     48838    (36.211, 40.053]
     48839    (55.421, 59.263]
     48840    (20.842, 24.684]
     48841    (51.579, 55.421]
     Name: age_bin, Length: 48784, dtype: category
     Categories (19, interval[float64, right]): [(16.927, 20.842]
     < (20.842, 24.684] < (24.684, 28.526] < (28.526, 32.368] ...
     (74.632, 78.474] < (78.474, 82.316] < (82.316, 86.158] < (8
     6.158, 90.0]]
```

```
plt.style.use('seaborn-ticks')
fig = plt.figure(figsize=(20,5))

plt.subplot(1, 2, 1)

sns.countplot(y="age_bin", data=data_num)

plt.subplot(1, 2, 2)

sns.distplot(data_num[data_num['income'] == '>50K']['age'], kde_kws={"label": ">$50K"})
sns.distplot(data_num[data_num['income'] == '<=50K']['age'], kde_kws={"label": "<=$50K"})
```

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



Hours of work

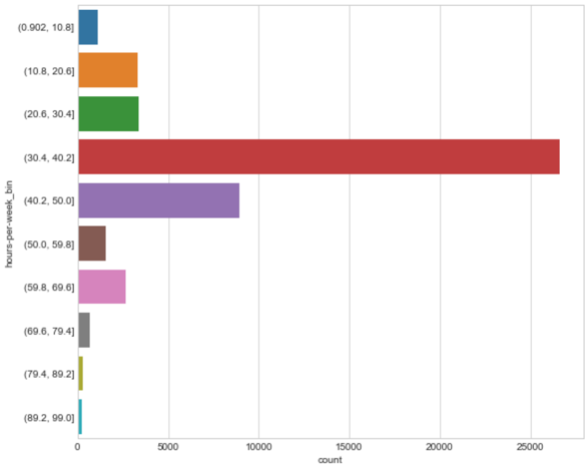
2]:

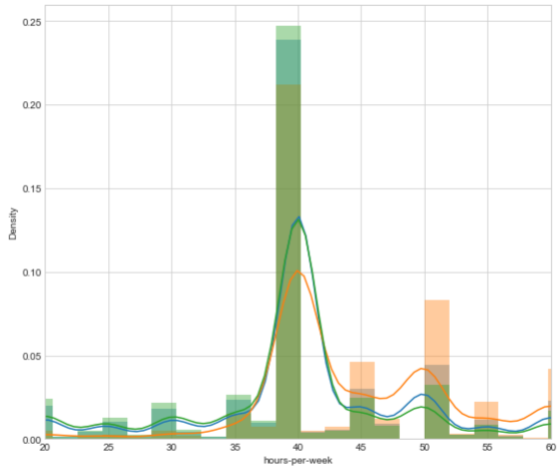
```
data_num['hours-per-week_bin'] = pd.cut(data_num['hours-per-week'], 10)
data_num['hours-per-week'] = data_num['hours-per-week']
```

3]:

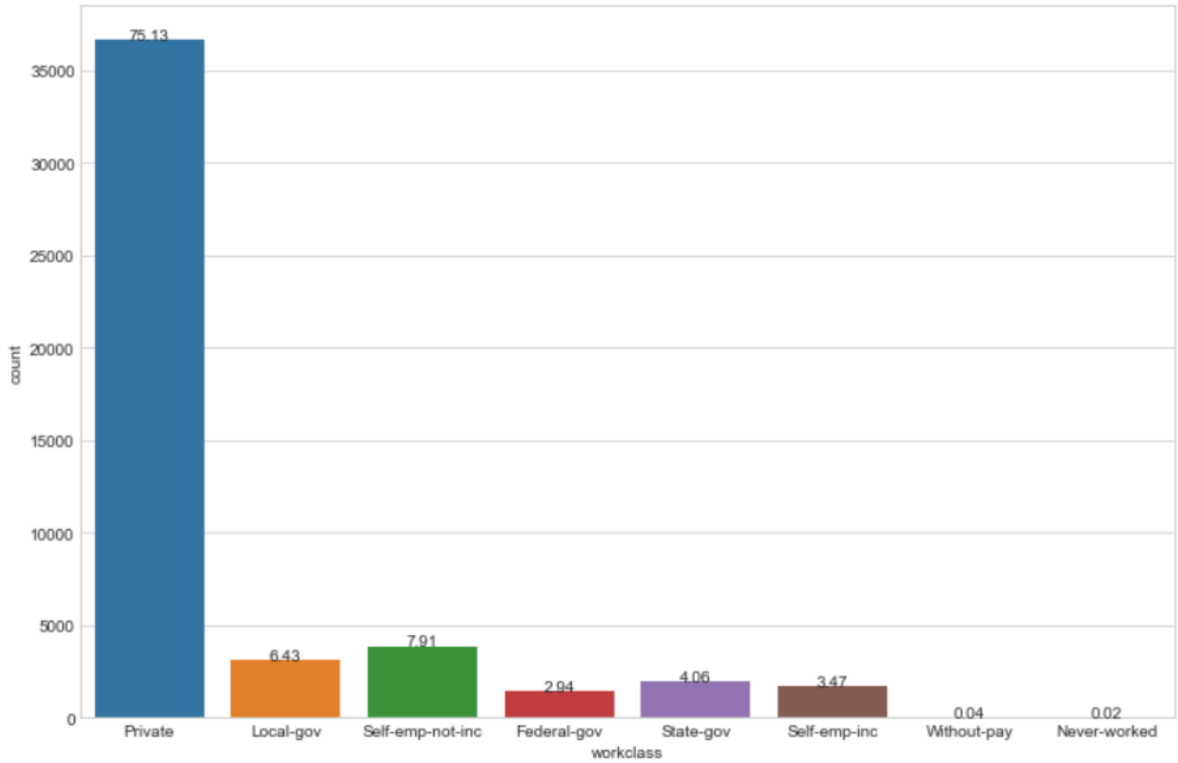
```
plt.style.use('seaborn-whitegrid')
fig = plt.figure(figsize=(20,8))

plt.subplot(1, 2, 1)
sns.countplot(y="hours-per-week_bin", data=data_num);

plt.subplot(1, 2, 2)

sns.distplot(data_num['hours-per-week']);
sns.distplot(data_num[data_num['income'] == '>50K']['hours-per-week'], kde_kws={"label": ">$50K"})
sns.distplot(data_num[data_num['income'] == '<=50K']['hours-per-week'], kde_kws={"label": "<$50K"})

plt.ylim(0, None)
plt.xlim(20, 60)
```

```
(20.0, 60.0)
```



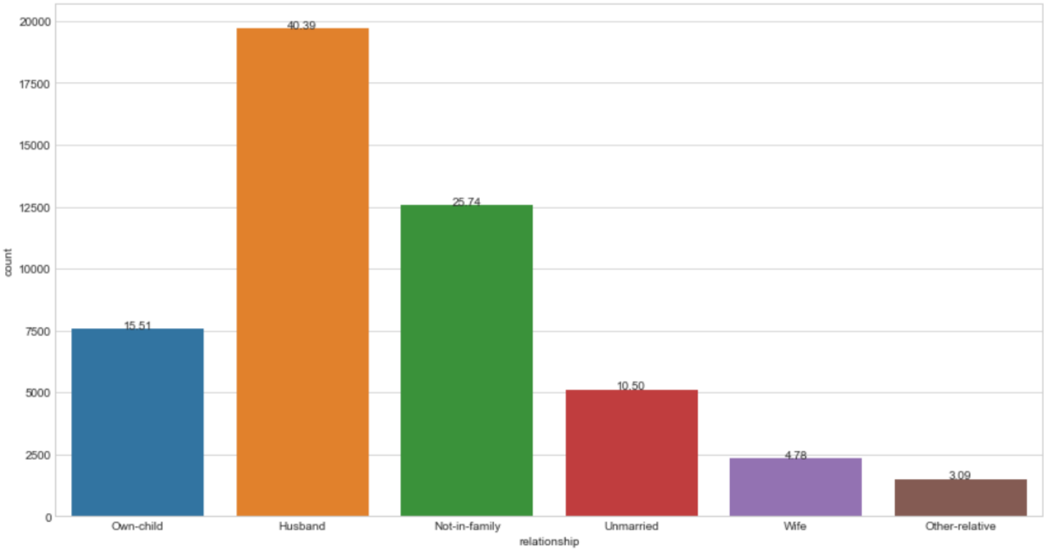


Work class



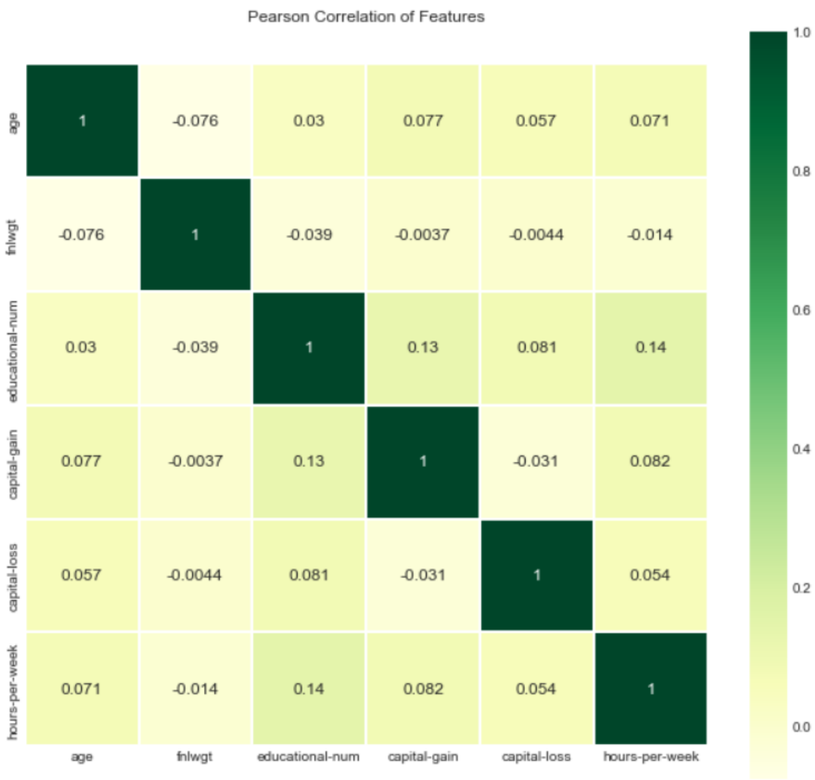
From the distribution plot, it is evident that there are 8 unique categories present in the work class attribute. Most of them belong to the private work class i.e., 75.13%. without-pay and never-worked have the minimum count in work class attribute (less than 1%).

Relationship



There are 6 unique categories in the relationship attribute. Husband has the maximum percentage (40.39%) among all categories followed by not-in-family (25.74%)

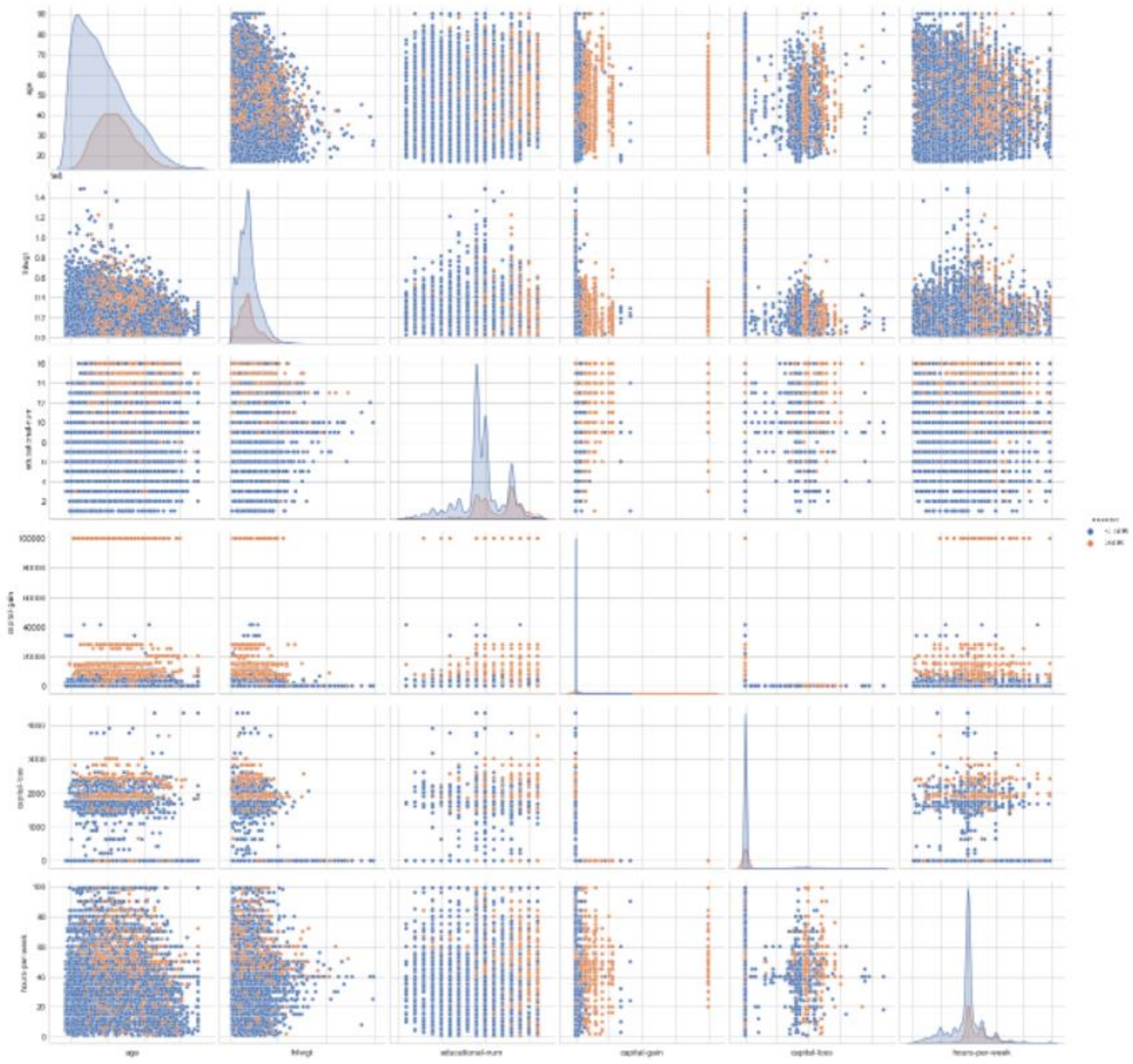
Correlation heatmap



There is neither strong positive nor strong negative correlation present in any variable. The strongest correlation is present between capital gain and hours-per-week with a Coefficient of 0.082.

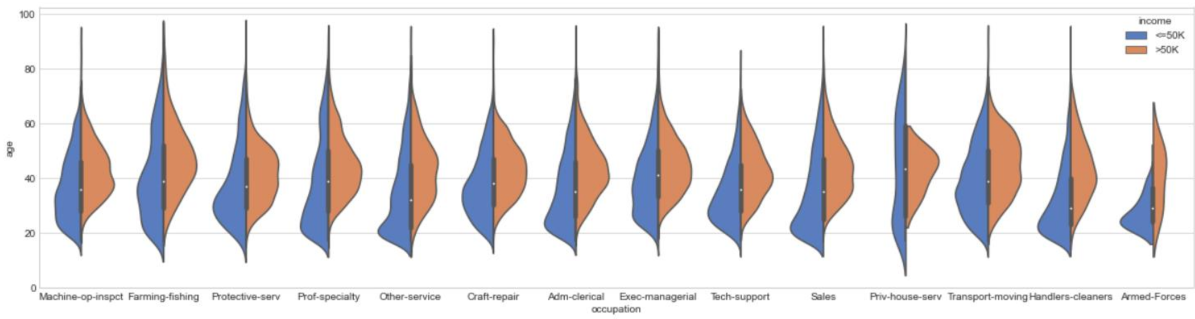
Pair plot from seaborn

<seaborn.axisgrid.PairGrid at 0x21f019a6340>

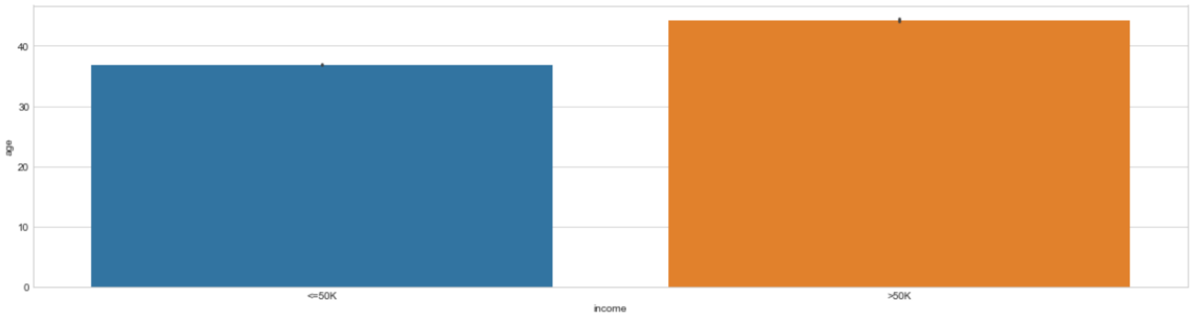


Occupation vs. Income Level

<AxesSubplot:xlabel='occupation', ylabel='age'>



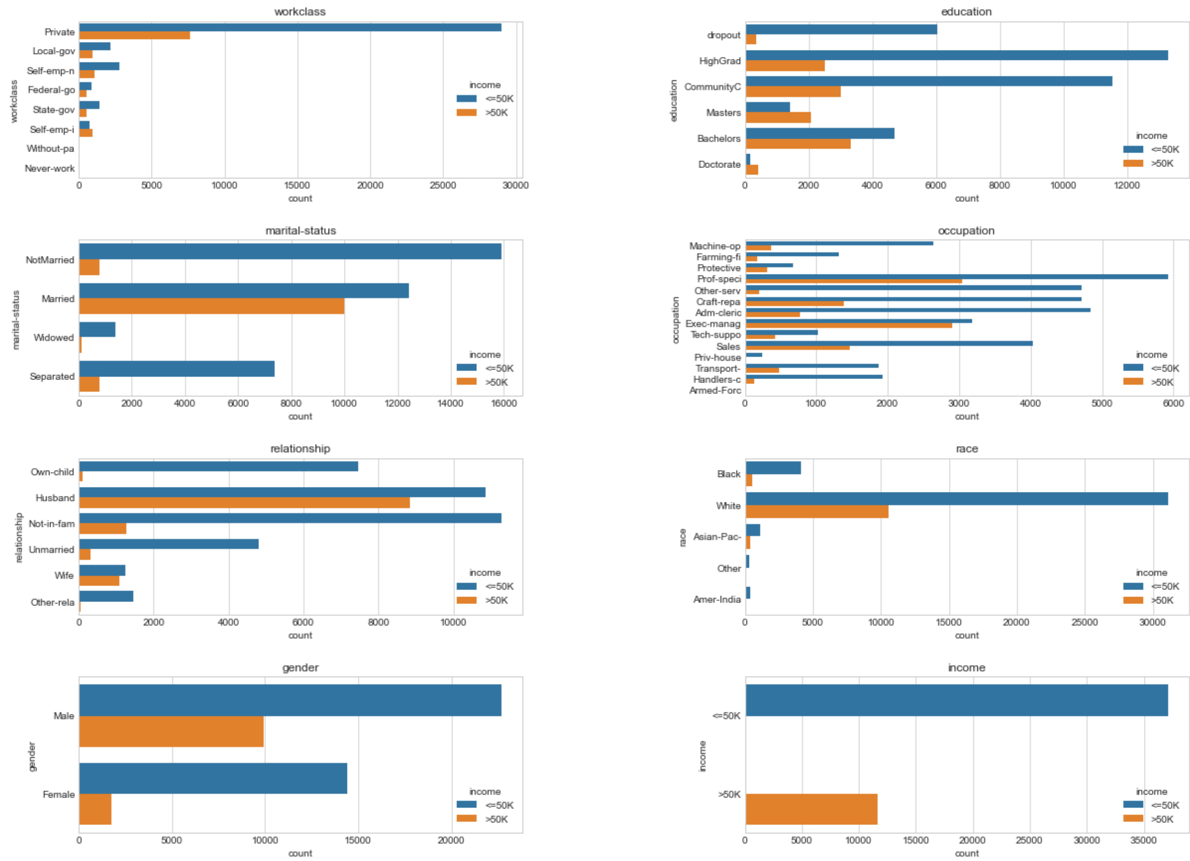
```
<AxesSubplot:xlabel='income', ylabel='age'>
```



The mean "age" for the Income group(<=50k) is 36.8 years. And for Income group(>50k) is 4 4.2year.

Bivariate Analysis

Bivariate analysis is one of the simplest forms of quantitative analysis, that involves the analysis of two variables, for the purpose of determining the empirical relationship between them.



Future Developments

- Since the basic data pre-processing steps are completed now the further steps like data splitting, data modeling, machine learning algorithm implementation and prediction are done.
- For the prediction we can perform classification using the following algorithms:
    - KNN (K-nearest neighbour)
    - Random Forest

Conclusion

- We have identified the dataset consists of some error data and performed the following the over-come it.
    - Incorrect Data – Dropping the Columns and Imputation (filling with most frequent attributes since string).
    - Duplicate Data – Identifies the duplicate the records and dropped the records.
- To identify the relationship between the features, we have created the correlation heat-map and performed the bivariant analysis.
    - We have identified the relationship between the age and income with density graph.
    - We have identified the attributes that are directly proportional using correlation matrix.

References

1. https://archive.ics.uci.edu/ml/datasets/adult
2. https://www.kaggle.com/code/pmarcelino/comprehensive-data-exploration-with-python/notebook
3. https://towardsdatascience.com /all-about-missing-data-handling-b94b8b5d2184
4. https://www.journaldev.com/53190/exploratory-data-analysis-python
5. https://medium.com/data-folks-indonesia/10-things-to-do-when-conducting-your-exploratory-data-analysis-eda-7e3b2dfbf812
6. https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python