NOTTINGHAM TRENT UNIVERSITY

School of Science and Technology

COMP20121
Machine Learning for Data Analysis Report

By
Bogyeong Kim
T0116478

1. Introduction (0.1, 1-2pg)

Analysis is the process of finding important patterns from data and conveying the interpreted outcomes from those patterns. To conduct a successful analysis proper planning is needed. CRISP-DM is one of the most frequently used methodologies for data mining analysis that in the world. CRISP-DM stands for 'Cross Industry Standard Process for Data Mining'. CRISP-DM breaks the process of data mining into six major phases:

1) Business understanding: In the first stage, it requires a preliminary plan to be made in order to understand the purpose of the project. This is done by using knowledge from a business perspective to define data mining problems.

2) Data understanding: The stage of data understanding begins to collect initial data, activities for data grasping and data quality checks to discover data insights and hidden information. Proceed from your home to the subset navigation you are interested in.

3) Data preparation: The data preparation stage covers all activities to construct the final dataset from the initial data. It seems that multiple tasks are to be performed in the data preparation task, but the order is not specified. Tasks include data conversion and tablet selection for modelling tools, as well as table, record, and attribute selection.

4) Modelling: In this step, various modelling methods are selected and applied. In general, there are several techniques for the same type of data mining problem. Some methods have specific requirements for the format of the data. In addition, there is often a need to return to the data preparation stage.

5) Evaluation: In this stage, the project creates a high-quality model from the viewpoint of data analysis. It is necessary to evaluate and review the initial model and perform the final stage: deployment.

6) Deployment: Generally, a model creation is not the end of a data mining project. If the purpose of the model is to expand the knowledge of the data, the acquired knowledge should be well organized and provide a way for customers to use it.

To apply CRISP-DM to this coursework well, one needs to understand the CRISP-DM methodology and then apply it accordingly. However, the descriptions of CRISP-DM have already been mentioned above, so it will not be explained again here. However, some detail about how this project is going to apply CRISP-DM will be explained below.

The first step for CRISP-DM is Business understanding. To understand this coursework, the survey dataset from Stack Overflow Developer Survey 2020 is used. This dataset is compiled of data collected from surveys about how they learn coding and level up, which tools they are using, and what they want. The purpose of this coursework is to find which features are relevant in regard to their income, whether low or high. It also needs to establish a plan, and that plan is: 1) arrange and organize the raw data; 2) research and choose which methods will be used for machine learning. 3) classify the data; and 4) analyse the data results and identify the features. When the plan is completed, it will proceed to the second step.

The second step is Data understanding. In this coursework, it needs to check and figure it out the data distribution in general. It will use python language and pandas library. When using this python language and library, it will check the data distribution by each column, and it will show graphs and tables for visualization. Through this process, the important columns and features will be found by comparing them with income.

The third step is data preparation. In this coursework, it needs to take the process for data refinement. When it is doing this process, it will use the relationship for the important features that already have finished to identify in the Business understanding step. Next, according to low income and high income, it needs to give labelling for the refinement data. Also, it needs to convert the data into numerical values. To apply for machine learning algorithms, it is necessary to separate the refinement data into a training set, validation set and test set.

The fourth step is Modelling. This coursework needs to research and classify algorithms from various kinds of methods. Through chosen algorithms, it needs to apply to prepared data, and fit this algorithm to the prepared data by changing them continuously. Finally, it will complete the final models by using continuous validation.

The fifth step is Evaluation. This coursework needs to apply the final model to the test set, and to evaluate the final model, it will use various "accuracy, sensitivity, specificity"' indicators. By analysing the results, it also needs to figure out which features have a high impact on income.

The sixth step is Deployment. In this coursework, when it has an additional survey, it collects the data, applies the algorithm and then evaluates the results. The results need to complement the final models consistently.

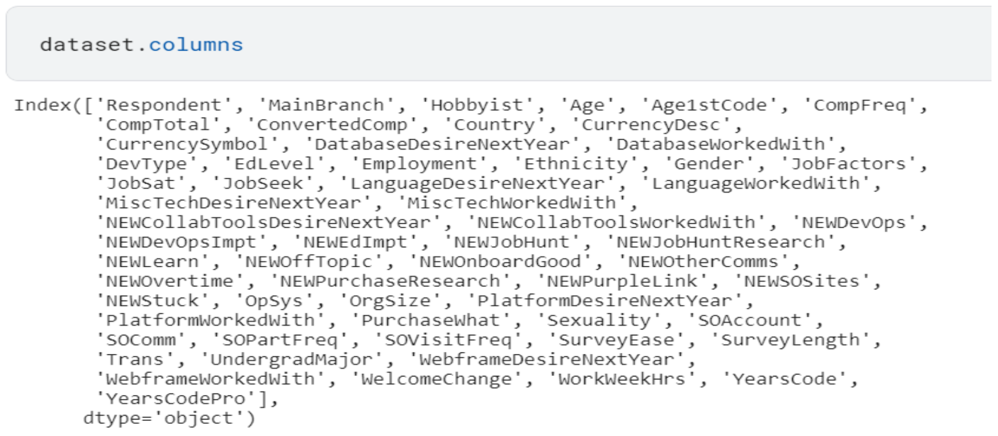
In brief, this project aims to answer the following questions:

1) What are the most important features for developers having higher salary?

2) Which classification techniques are most suitable for the prediction task and why are they better than others?

2. Data Understanding, Data Pre-processing, Exploratory Data Analysis

To describe the Survey Data Set, it collects the data from surveys about how they learn the coding and level up, which tools they are using, and what they want. This survey was taken in February 2020, and nearly 65,000 developers from 186 countries participated in this survey.
The dataset, has 61 columns and 64461 rows. It can be found from the code ***dataset.shape.*** Each columns (61 columns) name are same as like below picture, and it can be found ***dataset.columns*** code.



```
dataset.columns
```

```
Index(['Respondent', 'MainBranch', 'Hobbyist', 'Age', 'Age1stCode', 'CompFreq',
       'CompTotal', 'ConvertedComp', 'Country', 'CurrencyDesc',
       'CurrencySymbol', 'DatabaseDesireNextYear', 'DatabaseWorkedWith',
       'DevType', 'EdLevel', 'Employment', 'Ethnicity', 'Gender', 'JobFactors',
       'JobSat', 'JobSeek', 'LanguageDesireNextYear', 'LanguageWorkedWith',
       'MiscTechDesireNextYear', 'MiscTechWorkedWith',
       'NEWCollabToolsDesireNextYear', 'NEWCollabToolsWorkedWith', 'NEWDevOps',
       'NEWDevOpsImpt', 'NEWEdImpt', 'NEWJobHunt', 'NEWJobHuntResearch',
       'NEWLearn', 'NEWOffTopic', 'NEWOnboardGood', 'NEWOtherComms',
       'NEWOvertime', 'NEWPurchaseResearch', 'NEWPurpleLink', 'NEWSOSites',
       'NEWStuck', 'OpSys', 'OrgSize', 'PlatformDesireNextYear',
       'PlatformWorkedWith', 'PurchaseWhat', 'Sexuality', 'SOAccount',
       'SOComm', 'SOPartFreq', 'SOVisitFreq', 'SurveyEase', 'SurveyLength',
       'Trans', 'UndergradMajor', 'WebframeDesireNextYear',
       'WebframeWorkedWith', 'WelcomeChange', 'WorkWeekHrs', 'YearsCode',
       'YearsCodePro'],
      dtype='object')
```

Each Data columns' data type are different datatypes which are contained int64 (1), float64 (4) and object (56).

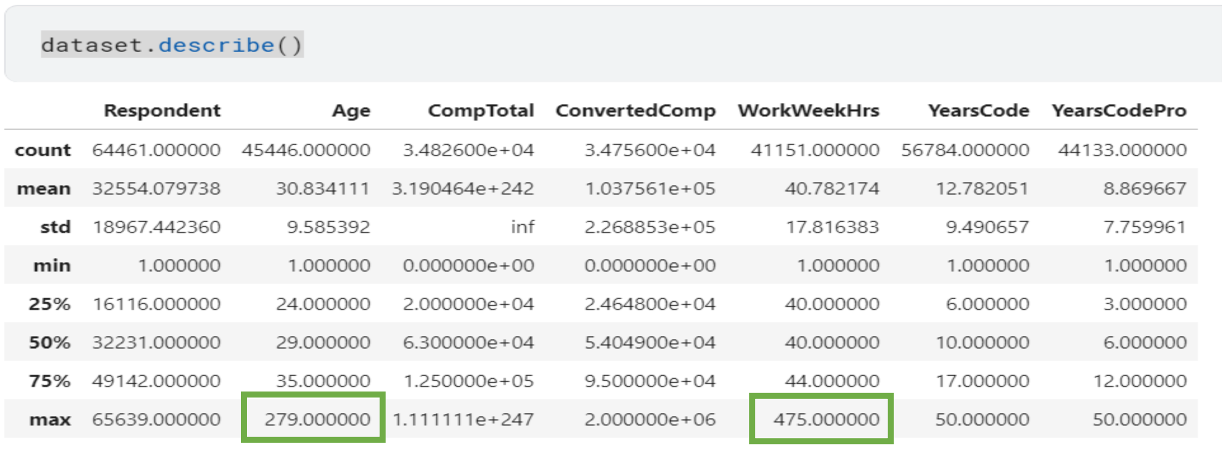
When you look at these 61 columns, they can be categorized. The survey consists of six sections: 1) Basic Information 2) Education, Work and Career 3) Technology and Tech Culture 4) Stack Overflow Usage + Community 5) Demographic Info 6) Final Qs. Therefore, the columns can be classified into six sections in accordance with the survey sections. The table below represents the result of the classified 61columns named in accordance with the six sections.

Sections	Columns Name
1.Basic Information	MainBranch, Hobbyist, Age, Gender, Sexuality, Trans
2.Education, Work and Career	1) Education: EdLevel, NEWEdImpt, UndergradMajor 2) Work: JobSat, NEWOvertime, NEWPurchaseResearch, OrgSize, WorkWeekHrs, CompFreq, CompTotal, ConvertedComp 3) Career: Age1stCode, DevType, Employment, JobFactors, JobSeek, NEWJobHunt, NEWJobHuntResearch, NEWLearn, YearsCode, YearsCodePro
3.Technology and Tech Culture	1) Technology: DatabaseDesireNextYear, DatabaseWorkedWith, LanguageDesireNextYear, LanguageWorkedWith, MiscTechDesireNextYear, MiscTechWorkedWith, NEWCollabToolsDesireNextYear, NEWCollabToolsWorkedWith, OpSys, PlatformDesireNextYear, PlatformWorkedWith, PurchaseWhat, WebframeDesireNextYear, WebframeWorkedWith, NEWPurpleLink, NEWStuck 2) Tech Culture: CurrencyDesc, CurrencySymbol
4.Stack Overflow Usage + Community	1) Stack Overflow Usage: NEWOffTopic, NEWSOSites, SOAccount, SOPartFreq, SOVisitFreq, WelcomeChange 2) Community: NEWOtherComms, SOComm
5.Demographic Info	Country, Ethnicity
6.Final Qs	SurveyEase, SurveyLength

To briefly do EDA (exploratory data analysis), firstly, the categorical EDA will be used.

This categorical EDA shows what are the top 5 values for each column. Also, it shows the Count for each columns of categories. It can easily process and categorize the data with an order of the top 5 values.   To analysis this survey dataset, most people who took this survey are from the USA and India. To see ethnicities, most people are White or of European descent, and the second largest percentages are South Asian people. For gender, most of the Respondents are men. Women percentages in this field are few. Most respondents are developers and full stacks. Most people either studied computer science or computer software and mostly graduated with a BA.

To describe the characteristics of the data, it can check and see the data in general by ***dataset.describe()*** code.



```
dataset.describe()
```

```
        Respondent           Age     CompTotal  ConvertedComp   WorkWeekHrs     YearsCode  YearsCodePro
count 64461.000000  45446.000000  3.482600e+04   3.475600e+04  41151.000000  56784.000000  44133.000000
mean  32554.079738     30.834111 3.190464e+242   1.037561e+05     40.782174     12.782051      8.869667
std   18967.442360      9.585392           inf   2.268853e+05     17.816383      9.490657      7.759961
min       1.000000      1.000000  0.000000e+00   0.000000e+00      1.000000      1.000000      1.000000
25%   16116.000000     24.000000  2.000000e+04   2.464800e+04     40.000000      6.000000      3.000000
50%   32231.000000     29.000000  6.300000e+04   5.404900e+04     40.000000     10.000000      6.000000
75%   49142.000000     35.000000  1.250000e+05   9.500000e+04     44.000000     17.000000     12.000000
max   65639.000000    279.000000 1.111111e+247   2.000000e+06    475.000000     50.000000     50.000000
```

From this code, it can check and recognize that there are outliers values: Age (max number: 279 age), WorkWeekHrs (max number: 475 hours, it is impossible to work on 67hours a day-> 475hours / 7days = 67.8 hours). Also, it can check missing values by ***dataset.isnull().sum*** code. There are many missing values in dataset, so It need the process of cleaning these data.

```
# Check the missing values
dataset.isnull().sum()
```

```
]:  Respondent                 0
    MainBranch               299
    Hobbyist                  45
    Age                    19015
    Age1stCode              6561
                           ...
    WebframeWorkedWith     22182
    WelcomeChange          11778
    WorkWeekHrs            23310
    YearsCode               7677
    YearsCodePro           20328
    Length: 61, dtype: int64
```

The last thing it needs to check is, duplicate values. These duplicate values can be found by ***dataset.duplicated().sum()*** code. The result of duplicate values is 0. So, this dataset doesn't need to eliminate the duplicated values.

```
dataset.duplicated().sum()
```

```
0
```

Therefore, we need to start the process of cleaning the data from outliers and missing values, which will ensure accurate results. These processes are demonstrated in the images below.

a. The process of cleaning outliers values: WorkWeekHrs, Age

```
notOutlier = dataset['WorkWeekHrs'] < 100
dataset = dataset[notOutlier]
```

```
dataset["WorkWeekHrs"]= dataset["WorkWeekHrs"][dataset["WorkWeekHrs"].between(dataset["WorkWeekHrs"].mean(),
                                                                                dataset["WorkWeekHrs"].quantile(.85)+20)]
```

```
dataset["WorkWeekHrs"].value_counts()
```

```
: 45.0    419
  50.0    295
  60.0    110
  42.0    101
  48.0     56
  44.0     53
  55.0     47
  43.0     37
  46.0     12
  56.0     11
  47.0     11
  54.0      8
  65.0      6
  49.0      5
  52.0      4
  42.5      3
  63.0      2
  51.0      1
  42.4      1
  53.0      1
  58.0      1
  62.0      1
  57.0      1
  Name: WorkWeekHrs, dtype: int64
```

b. The process of cleaning missing values:

```python
# Drop the values '0' with 61 columns
dataset.dropna(inplace=True)
dataset.head(20)
```

After eliminating the outliers values and missing values, the dataset is changed. The dataset becomes more accurate and dataset shape also gets reduced from (64461 rows, 61columns) to (1206 rows, 61 columns).

```python
dataset.describe()
```

	Respondent	Age	CompTotal	ConvertedComp	WorkWeekHrs	YearsCode	YearsCodePro
count	1206.000000	1206.000000	1.206000e+03	1.206000e+03	1206.000000	1206.000000	1206.000000
mean	28689.154229	32.805970	3.006853e+06	1.254721e+05	48.092786	15.880597	9.925373
std	18723.697248	7.770601	2.900900e+07	2.769719e+05	5.257654	9.000821	7.322522
min	119.000000	18.000000	0.000000e+00	0.000000e+00	41.000000	2.000000	1.000000
25%	12287.250000	27.000000	3.800000e+04	2.565600e+04	45.000000	9.000000	4.000000
50%	26435.500000	31.000000	9.740000e+04	6.892200e+04	45.000000	14.000000	8.000000
75%	47361.250000	37.000000	1.600000e+05	1.200000e+05	50.000000	20.000000	14.000000
max	62904.000000	61.000000	9.000000e+08	2.000000e+06	65.000000	45.000000	38.000000

```python
dataset.shape
```

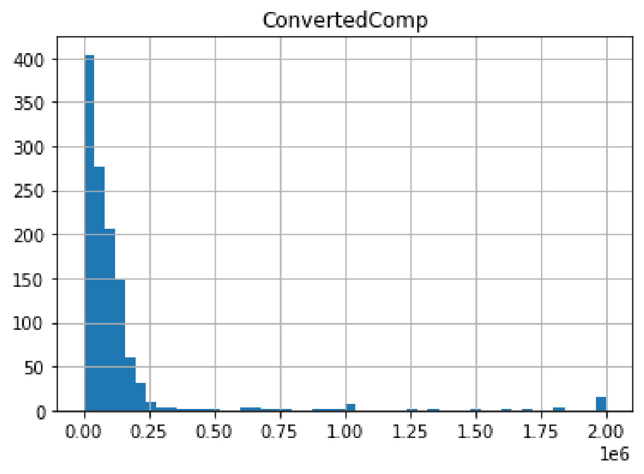
```
(1206, 61)
```

To conduct EDA on the data set, first it uses histogram. Histograms is a pictorial representation of a set of data. It is created by grouping the measurements into "cells "or "bins". Histograms are simplicity and versatility. The one of benefits of histogram is organizations. It supports finding and dealing with process variation quickly. (TQP, March 17,2020)

Figure1 : EDA- histogram(ConvertedComp)

```python
dataset.hist(column="ConvertedComp",bins=50)
```

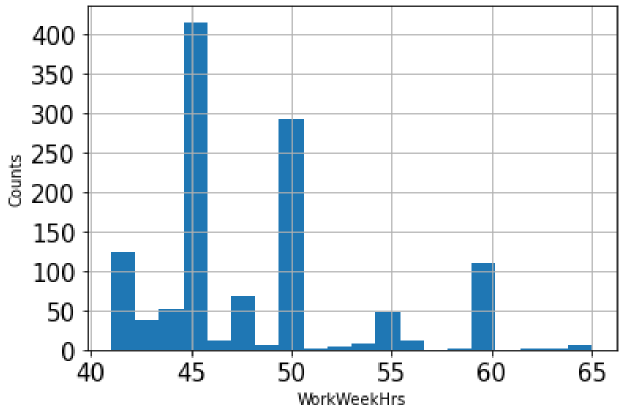
```
array([[<AxesSubplot:title={'center':'ConvertedComp'}>]], dtype=object)
```



In this figure1, it is normally concentrated on 0~250000 low income area. This is a target value, so it is important data. If using simple thresholding for labeling, it can bring out the result of underfitting (hard to study for machine learning classification) or overfitting (study by unequally distributed). It is because the distribution is non-uniform. Therefore, it needs to get more target value data or need additional analysis using clustering methods and then careful labeling.

Figure2: EDA- histogram(WorkWeekHrs)

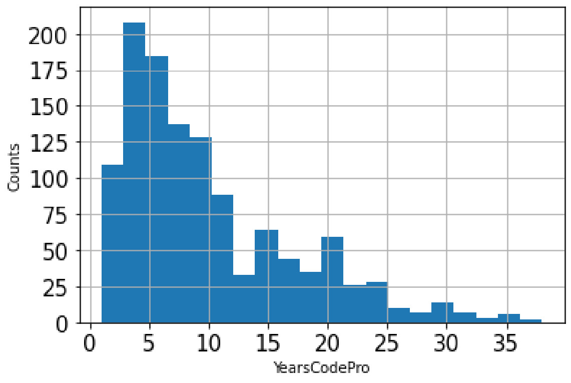
```
import matplotlib.pyplot as plt
plt.figure()
dataset['WorkWeekHrs'].plot(kind='hist',grid=True, bins=20,fontsize=15)
plt.xlabel('WorkWeekHrs')
plt.ylabel('Counts')
plt.show()
```



In this figure2, overall distribution is non-uniform. Mostly people work around 45 hours. Which means people work around 9hours a day(45h/5d).

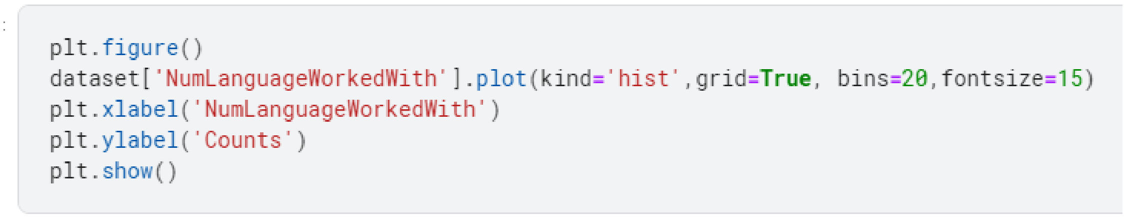
Figure3: EDA- histogram(YearsCodePro)

```
plt.figure()
dataset['YearsCodePro'].plot(kind='hist',grid=True, bins=20,fontsize=15)
plt.xlabel('YearsCodePro')
plt.ylabel('Counts')
plt.show()
```

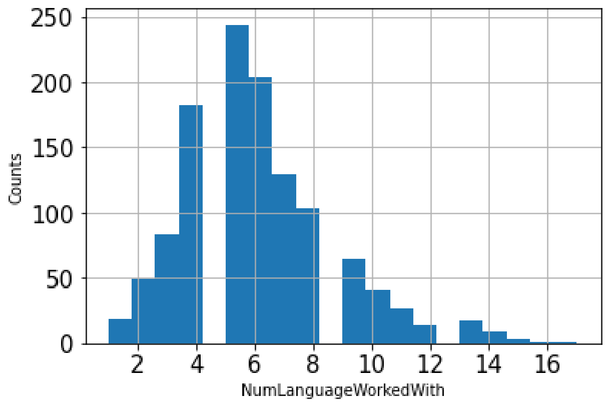


In this figure3, it shows that most people work for code as professional 1~15 years. Also, overall, the number of people for coding as professional is decreasing. Therefore, it would be better put some weight for better accuracy as the years code as professional is increasing.

Figure4: EDA- histogram(NumLanguageWorkedWith)

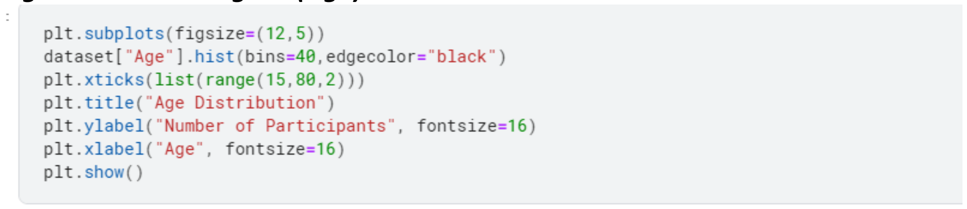


```
plt.figure()
dataset['NumLanguageWorkedWith'].plot(kind='hist',grid=True, bins=20,fontsize=15)
plt.xlabel('NumLanguageWorkedWith')
plt.ylabel('Counts')
plt.show()
```

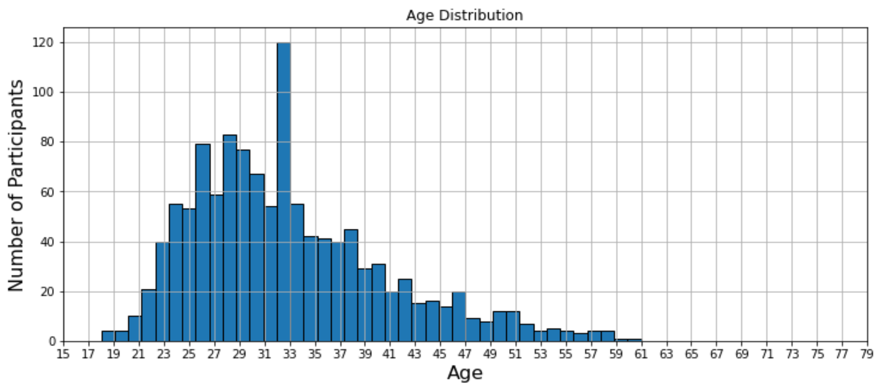


‘NumLanguageWorkedWith’ column is a new created column from raw data. This column means how many numbers of language people use when they work. In this figure4, it is also non-uniform. So, it can be distorted the result of accuracy.

Figure5: EDA- histogram(Age)



```
plt.subplots(figsize=(12,5))
dataset["Age"].hist(bins=40,edgecolor="black")
plt.xticks(list(range(15,80,2)))
plt.title("Age Distribution")
plt.ylabel("Number of Participants", fontsize=16)
plt.xlabel("Age", fontsize=16)
plt.show()
```



In this figure5, it shows age distribution. Most respondent are mainly 20-39 ages.

To summarize for histogram, histogram has benefits it can check the overall density distribution. Therefore, it is important to check data uniformity and data regularity. If the data is non-uniform, it needs to convert to categorical data from continuous data. Also, if the data has less density relatively, it needs to give some weight. This will prevent the further overfitting or underfitting state. Also, if the data can be found the certain rules, it can make the data rearrange or choose proper learning method.

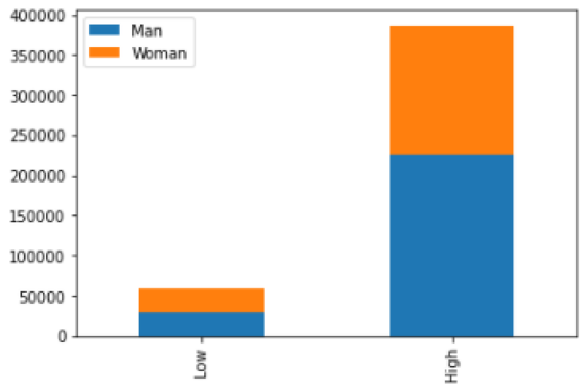
Second way for EDA is crosstab. A key benefit of the crosstab is that it allows to normalize the resulting data frame, returning values displayed as percentages. (Kumar, March 5,2021)

```
salary_median = dataset['ConvertedComp'].median()
y = dataset['ConvertedComp'].apply(lambda x:0 if x <= salary_median else 1)
dataset['Income'] = y
dataset.head(20)
```

Figure6: EDA- crosstab(Gender)

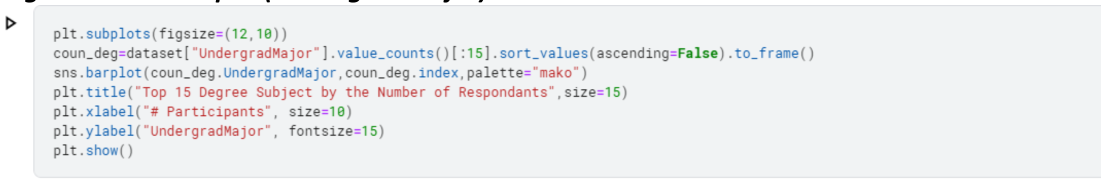
```
genders = dataset['Gender'].drop_duplicates()
incomes = np.zeros(len(dataset['Income'].drop_duplicates()))
my_test = [];
for num, gender in enumerate(genders):
    filter0 = (dataset['Gender'] == gender) & (dataset['Income'] == 0)
    filter1 = (dataset['Gender'] == gender) & (dataset['Income'] == 1)
    values0 = dataset[filter0].loc[:, 'ConvertedComp']
    values1 = dataset[filter1].loc[:, 'ConvertedComp']
    my_test.append([values0.mean(), values1.mean()])

print(my_test)

my_test = np.transpose(np.asarray(my_test))
my_test = pd.DataFrame(data=my_test, columns=['Man','Woman'], index=['Low','High'])
print('crosstab for numerical features:')
print(my_test)
#fig, ax = plt.subplots(figsize=(10,7))
my_test.plot.bar(stacked=True)
plt.show()
```

```
[[29583.331615120274, 225527.79649122807], [29101.533333333333, 161353.57692307694]]
crosstab for numerical features:
                Man          Woman
Low     29583.331615   29101.533333
High   225527.796491  161353.576923
```

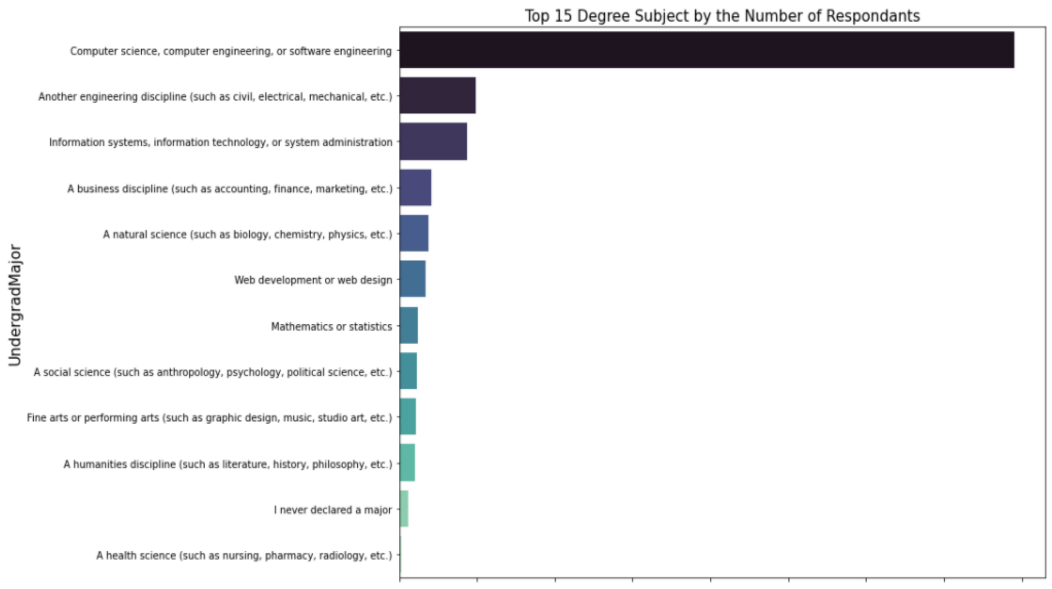


In this figure6, it is crosstab. It shows the percentages of Gender (Man, Woman) by Low income & Hight income. In Low income, it has similar percentages by Man and Woman. In High income, it looks like Man percentages is bigger than the Woman percentages.

Figure7: EDA- barplot(UndergradMajor)

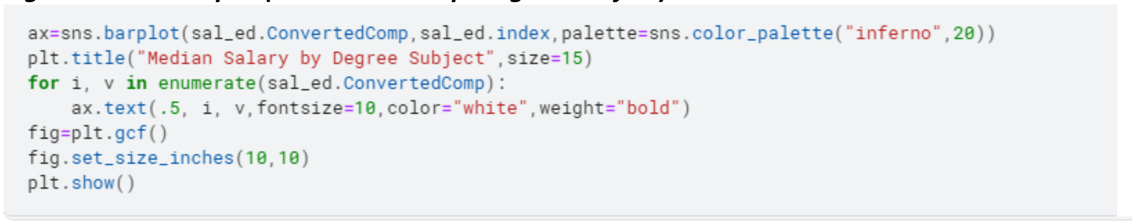


```
plt.subplots(figsize=(12,10))
coun_deg=dataset["UndergradMajor"].value_counts()[:15].sort_values(ascending=False).to_frame()
sns.barplot(coun_deg.UndergradMajor,coun_deg.index,palette="mako")
plt.title("Top 15 Degree Subject by the Number of Respondants",size=15)
plt.xlabel("# Participants", size=10)
plt.ylabel("UndergradMajor", fontsize=15)
plt.show()
```

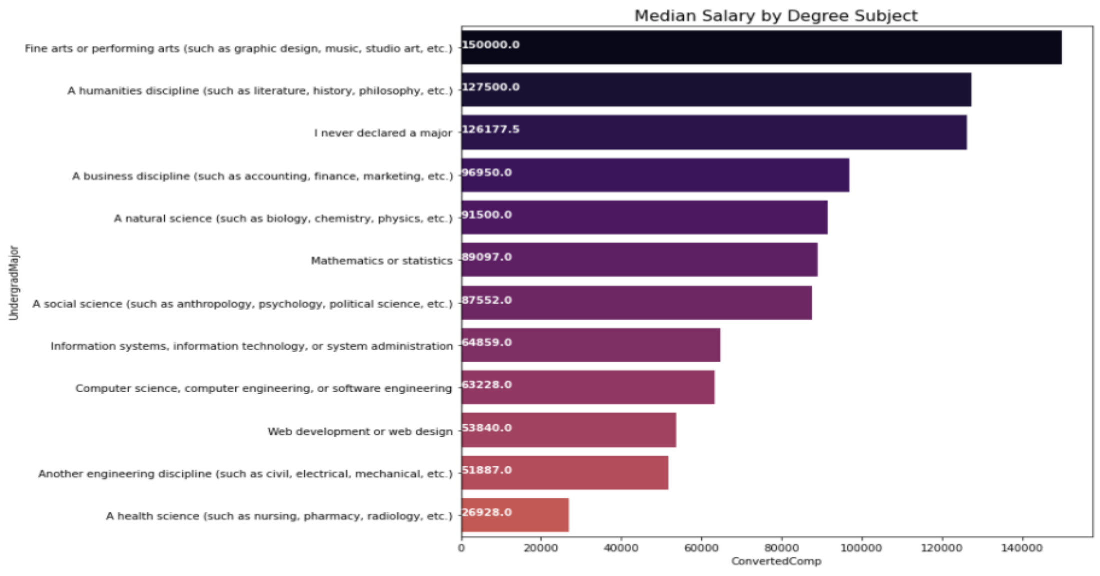


In this figure7, the most respondent's degree major is Computer science, computer engineering or software engineering. From this result, it can guess who worked in IT field, they also studied their major as IT: computer science, computer engineering or software engineering.

Figure8: EDA- barplot(ConvertedComp-Degree Subject)



```
ax=sns.barplot(sal_ed.ConvertedComp,sal_ed.index,palette=sns.color_palette("inferno",20))
plt.title("Median Salary by Degree Subject",size=15)
for i, v in enumerate(sal_ed.ConvertedComp):
    ax.text(.5, i, v,fontsize=10,color="white",weight="bold")
fig=plt.gcf()
fig.set_size_inches(10,10)
plt.show()
```



In this figure8, it shows something unexpected result. In my expectation, I thought related with IT major will be the high Median Salary. However, in this result shows that who studied Fine Arts or performing arts (such as graphic design, music, studio art, etc.) major is highest median Salary.

3. Cluster Analysis

Clustering is one of the most common exploratory data analysis techniques used to get an intuition about the structure of the data. (Dabbura, Sep 18, 2018)

In this project for cluster analysis, it is better to split the data set into two subsets, labelled low-income and high-income. In this project, income is given in the column **ConvertedComp**. To split the ConvertedComp to low-income or high-income, first it needs to calculate **median()** for ConvertedComp. After then, it will use **lambda** function in python: (if x is smaller than salary_median then it will return 0, if x is bigger than salary_median then it will return 1) to split ConvertedComp into 0(low-income) or 1(high-income). It shows the values counts of “Income” columns: 0 (low-income): 597, 1(high-income): 596.

```
# Salary_median
salary_median = dataset['ConvertedComp'].median()

# 1: for High Income / 0: for Low Income
y = dataset['ConvertedComp'].apply(lambda x: 0 if x <= salary_median else 1)

dataset['Income'] = y

dataset.head(10)
```

```
dataset["Income"].value_counts()
```

```
0    597
1    596
Name: Income, dtype: int64
```

In this project K-Means clustering methods will be used. K-Means is considered as one of the most used clustering algorithms because of its simplicity. K-Means clustering algorithm is just two steps: the cluster assignment step, and the move centroid step. It is easy to implement and can handle large data sets. This method is used to group unlabeled data set instances into clusters based on similar attributes. It can cluster the data sets just using the number of clusters and then does not require the labelling. (automaticaddison, July 23, 2019)

Figure9: K-Means clustering (ConvertedComp-income)

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 2, random_state = 0).fit(ConvertedComp.reshape(-1,1))

import numpy as np
import matplotlib.pyplot as plt
print(kmeans.cluster_centers_)
```

```
[[  81900.26165803]
 [1587415.88571429]]
```

```
# Using K-Means clustering method

labels_1 = np.where(kmeans.labels_ == 0)
labels_2 = np.where(kmeans.labels_ == 1)

print(labels_1[0])
print(labels_2[0])

plt.scatter(ConvertedComp[labels_1], np.ones((len(ConvertedComp[labels_1]),)).tolist(), s = 5.0)
plt.scatter(ConvertedComp[labels_2], np.ones((len(ConvertedComp[labels_2]),)).tolist(), s = 5.0)
plt.title('incomes_clutered')
plt.show()
```

```
[   0    1    2 ... 1190 1191 1192]
[  29   82  111  114  130  151  165  204  206  264  272  351  354  424
  462  468  477  557  561  642  655  669  673  757  787  830  865  997
 1024 1027 1029 1058 1092 1095 1163]
```

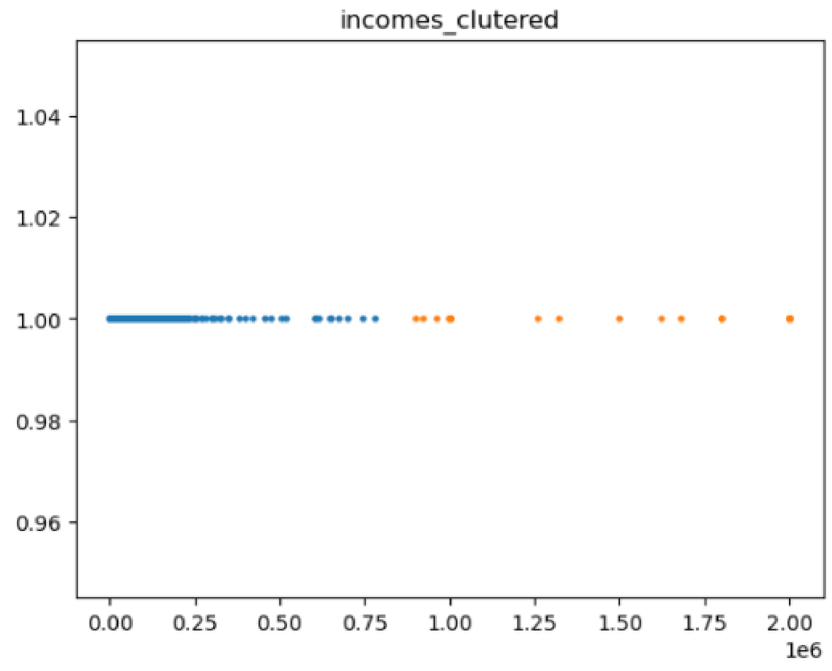
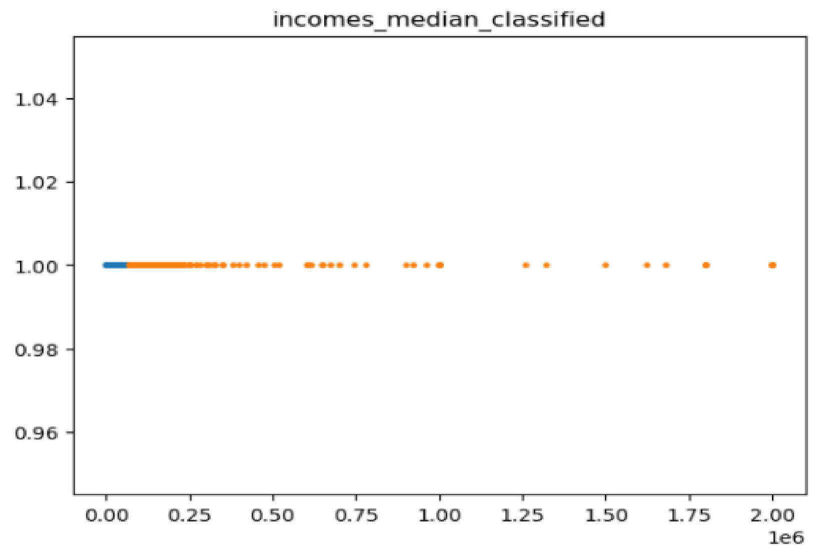


Figure10: using Data Mean - thresholding classify (ConvertedComp-income)

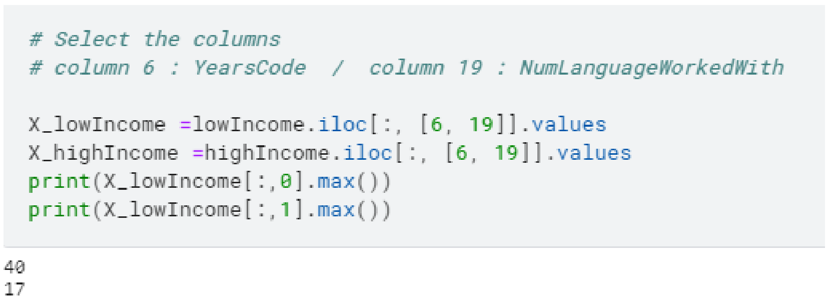
```
#The graph that classify to use thressholding of Data means

label_1 = np.where(y.values == 0)
label_2 = np.where(y.values == 1)

plt.scatter(ConvertedComp[label_1], np.ones((len(y.values[label_1]),)).tolist(), s = 5.0)
plt.scatter(ConvertedComp[label_2], np.ones((len(y.values[label_2]),)).tolist(), s = 5.0)
plt.title('incomes_median_classified')
plt.show()
```



To compare figure 9 and 10, figure 9 (K-Means clustering methods) does not get affected by the imbalanced amount of data. But It does get affected by the data values. However, the figure 10 (using data mean – thresholding classifies methods) gets affected by the data amount. Figure 10 are crowded around low income and breakpoint is more low-income side. When the breakpoint is lopsided towards one side, it can bring out the generalization problems. However, Figure 9 (K-Means clustering methods) looks like more evenly spread and the breakpoint does not lopsided towards one side. It is on almost at the middle of the sides.

To describe data transformation and normalization, firstly, unnecessary values need to be eliminated (NaN values, duplicated values, outlier values, etc.). After then, machine learning is based on the number values, so it needs to convert string values into numerical values. Finally, the deviation of columns' values is big. Therefore, it needs to undertake the normalization process, which converts the columns' values into 0~1 value.

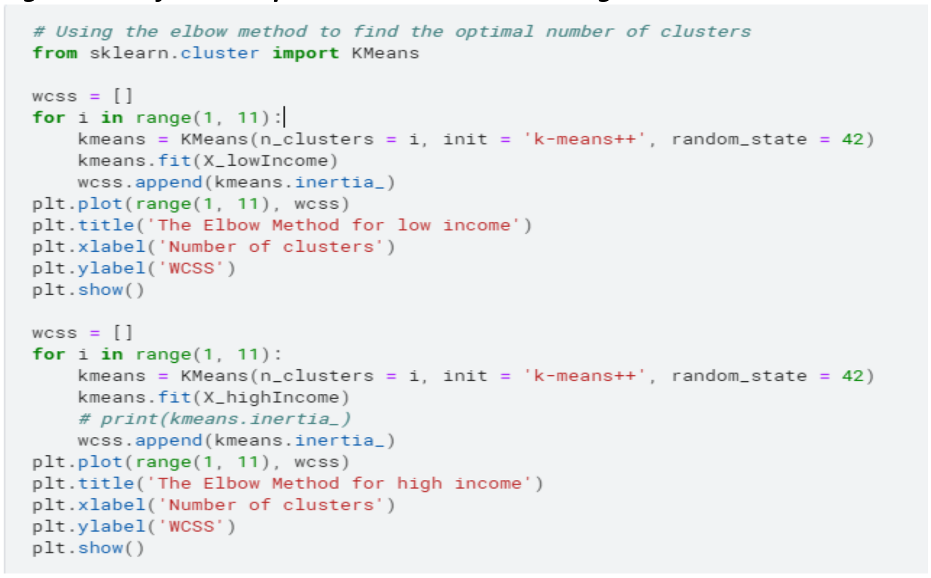
To implement cluster analysis on each subset, I choose the 2 columns which are 'YearsCode' and 'NumLanguageWorkedWith', because they were numerical values and looked relative with salary. Also, I used the **Elbow method** to find the optimal cluster number. K-Means is unsupervised machine learning algorithm that groups data into k number of clusters. The number of clusters is user-defined so it is important to choose optimal cluster number. The Elbow method is a very popular technique to run k-means clustering for a range of clusters k. To find the right cluster number is, when the plot looks like an arm then the "elbow" (the point of inflection on the curve) is the best value of k. (Ersi, Auguest 1, 2020)



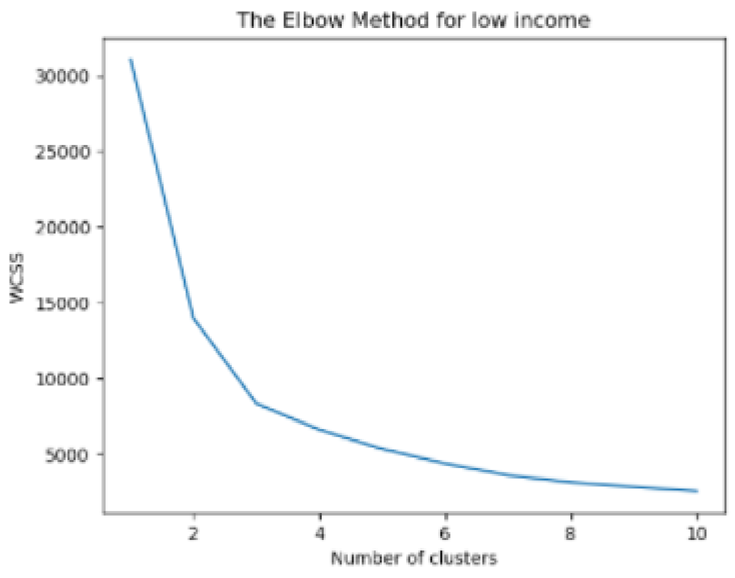
```
# Select the columns
# column 6 : YearsCode  /  column 19 : NumLanguageWorkedWith

X_lowIncome =lowIncome.iloc[:, [6, 19]].values
X_highIncome =highIncome.iloc[:, [6, 19]].values
print(X_lowIncome[:,0].max())
print(X_lowIncome[:,1].max())
```

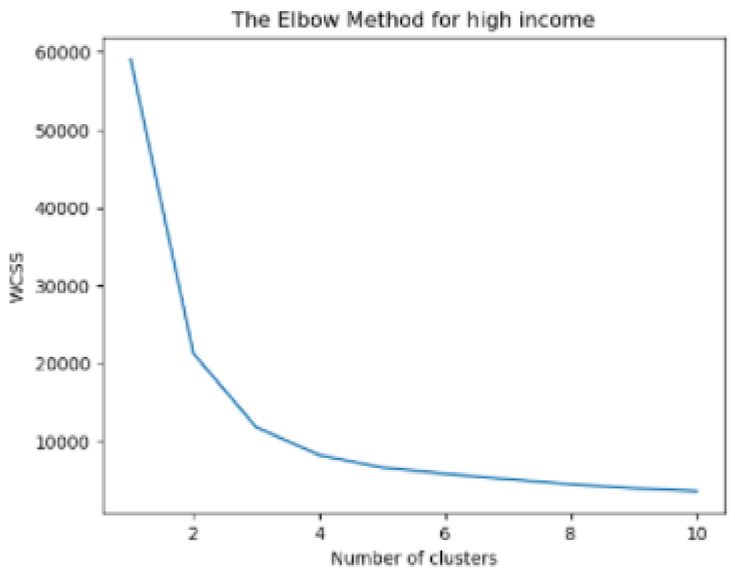
```
40
17
```

Figure11: To find the optimal cluster number using Elbow method.



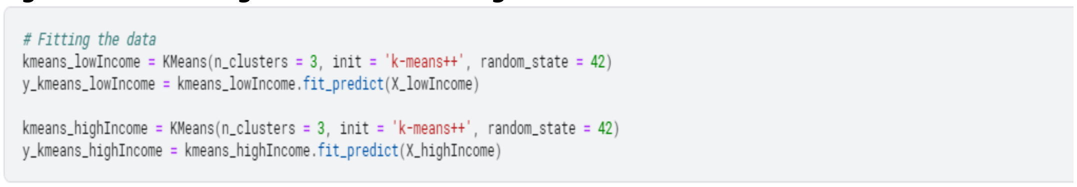
```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X_lowIncome)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method for low income')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X_highIncome)
    # print(kmeans.inertia_)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method for high income')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



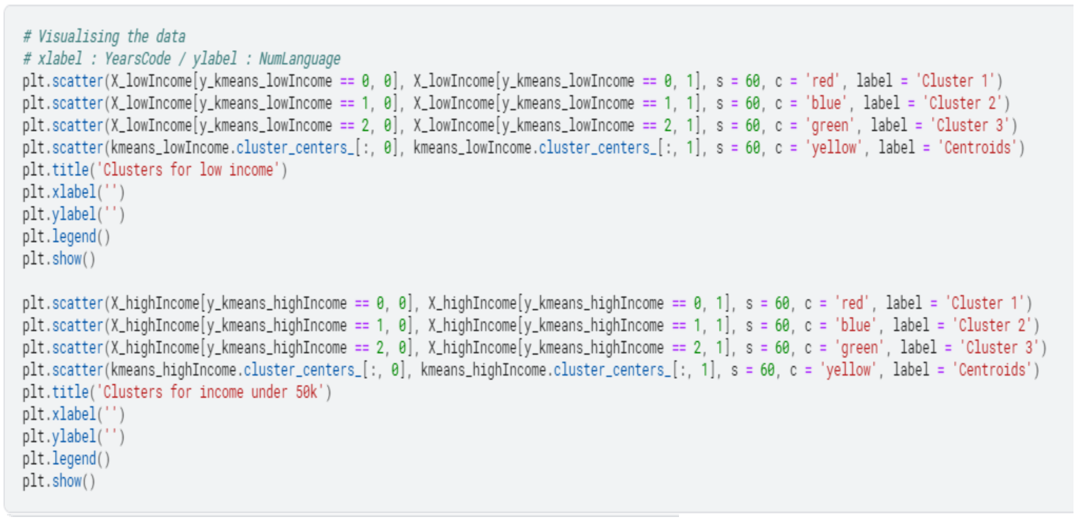


When you look at this figure 11, it can be found number 3 is “elbow” (the point of inflection on the curve). It means number 3 is the best values of k. Therefore, throughout this process, I choose the cluster number “**3**”.

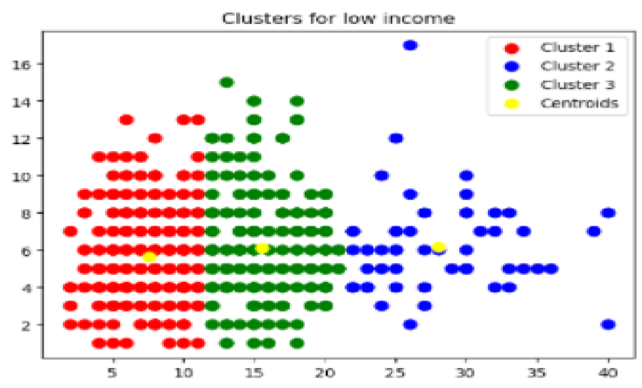
Figure12: Visualizing data to cluster using cluster number "3"

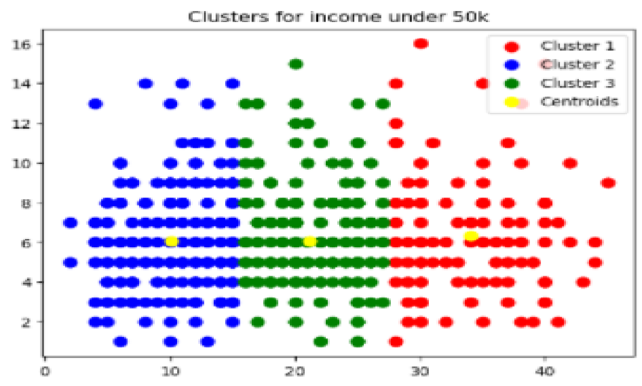


```
# Fitting the data
kmeans_lowIncome = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans_lowIncome = kmeans_lowIncome.fit_predict(X_lowIncome)

kmeans_highIncome = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans_highIncome = kmeans_highIncome.fit_predict(X_highIncome)
```



```
# Visualising the data
# xlabel : YearsCode / ylabel : NumLanguage
plt.scatter(X_lowIncome[y_kmeans_lowIncome == 0, 0], X_lowIncome[y_kmeans_lowIncome == 0, 1], s = 60, c = 'red', label = 'Cluster 1')
plt.scatter(X_lowIncome[y_kmeans_lowIncome == 1, 0], X_lowIncome[y_kmeans_lowIncome == 1, 1], s = 60, c = 'blue', label = 'Cluster 2')
plt.scatter(X_lowIncome[y_kmeans_lowIncome == 2, 0], X_lowIncome[y_kmeans_lowIncome == 2, 1], s = 60, c = 'green', label = 'Cluster 3')
plt.scatter(kmeans_lowIncome.cluster_centers_[:, 0], kmeans_lowIncome.cluster_centers_[:, 1], s = 60, c = 'yellow', label = 'Centroids')
plt.title('Clusters for low income')
plt.xlabel('')
plt.ylabel('')
plt.legend()
plt.show()

plt.scatter(X_highIncome[y_kmeans_highIncome == 0, 0], X_highIncome[y_kmeans_highIncome == 0, 1], s = 60, c = 'red', label = 'Cluster 1')
plt.scatter(X_highIncome[y_kmeans_highIncome == 1, 0], X_highIncome[y_kmeans_highIncome == 1, 1], s = 60, c = 'blue', label = 'Cluster 2')
plt.scatter(X_highIncome[y_kmeans_highIncome == 2, 0], X_highIncome[y_kmeans_highIncome == 2, 1], s = 60, c = 'green', label = 'Cluster 3')
plt.scatter(kmeans_highIncome.cluster_centers_[:, 0], kmeans_highIncome.cluster_centers_[:, 1], s = 60, c = 'yellow', label = 'Centroids')
plt.title('Clusters for income under 50k')
plt.xlabel('')
plt.ylabel('')
plt.legend()
plt.show()
```





This figure 12, it is the result of the visualization clustering with cluster number '3' (**x-axis** : 'YearsCode' and **y-axis** : 'NumLanguageWorkedWith'). Also, red color represents cluster1, blue is for cluster2 and green is for cluster3. Yellow color is for centroids of each clustering.

To describe the k-Means parameter setting, it uses **Euclidean distance** and the number of clusters is **3.** Also, the **Max_iter** is 300 times. **N_init** (number of times the k-Means algorithm will be run with different centroid seeds) is 10 times.

Describing characteristics of low-income developer clusters and high-income developer clusters, here is a table about summary of characteristics of low/high income.

Columns	Low income	High income	Low vs High income
Age	For each cluster, mostly there are 10s and 20s young people, and there is less difference between each cluster.	Each cluster, mostly 10s and 20s young people are present, but it shows that 30s and 40s people is increasing. Also, to see the difference between clusters, cluster1 seems like having more younger people than the other two clusters.	It shows that high income people's age is about 5~6 years older than the low-income developer.
Age1stCode	Most developers start to write code about under 10 years or 10s.	Most developers write first time code is also under 10 years or 10s.	It shows that high income developers start writing 1st time code around 2~3 years. They start to write code earlier than the low-income developer.
YearsCode	Overall, it shows that years of code are on from 10 years to 20years. The average of each cluster yearsCode is about 10years.	Overall, they are more widely distributed for 40 years.	High income developer is having more experiences about 6~8years than low-income developer.
YearsCodePro	Overall, YearsCodePro is distributed to until 30 years. However, mostly it distributed within 10 years.	Overall, YearsCodePro is also distributed to until 30 years. However, mostly it distributed 10~20 years.	
Country	It is distributed to various countries. In relatively, for cluster0, cluster2 is distributed to India, However, cluster1 is usually distributed to United Kingdom.	Most people are in United States for high-income developers.	It shows that most people for high-income developers are in United States.
Jobsatisfy	Most people feel satisfied with 3 rate (out of 4 range: 4 is the high satisfy rate).	It shows mostly people are satisfied with 4 (the most high satisfy rate)	

Figure13: Low income - Visualizing data with cluster1~3 & each columns

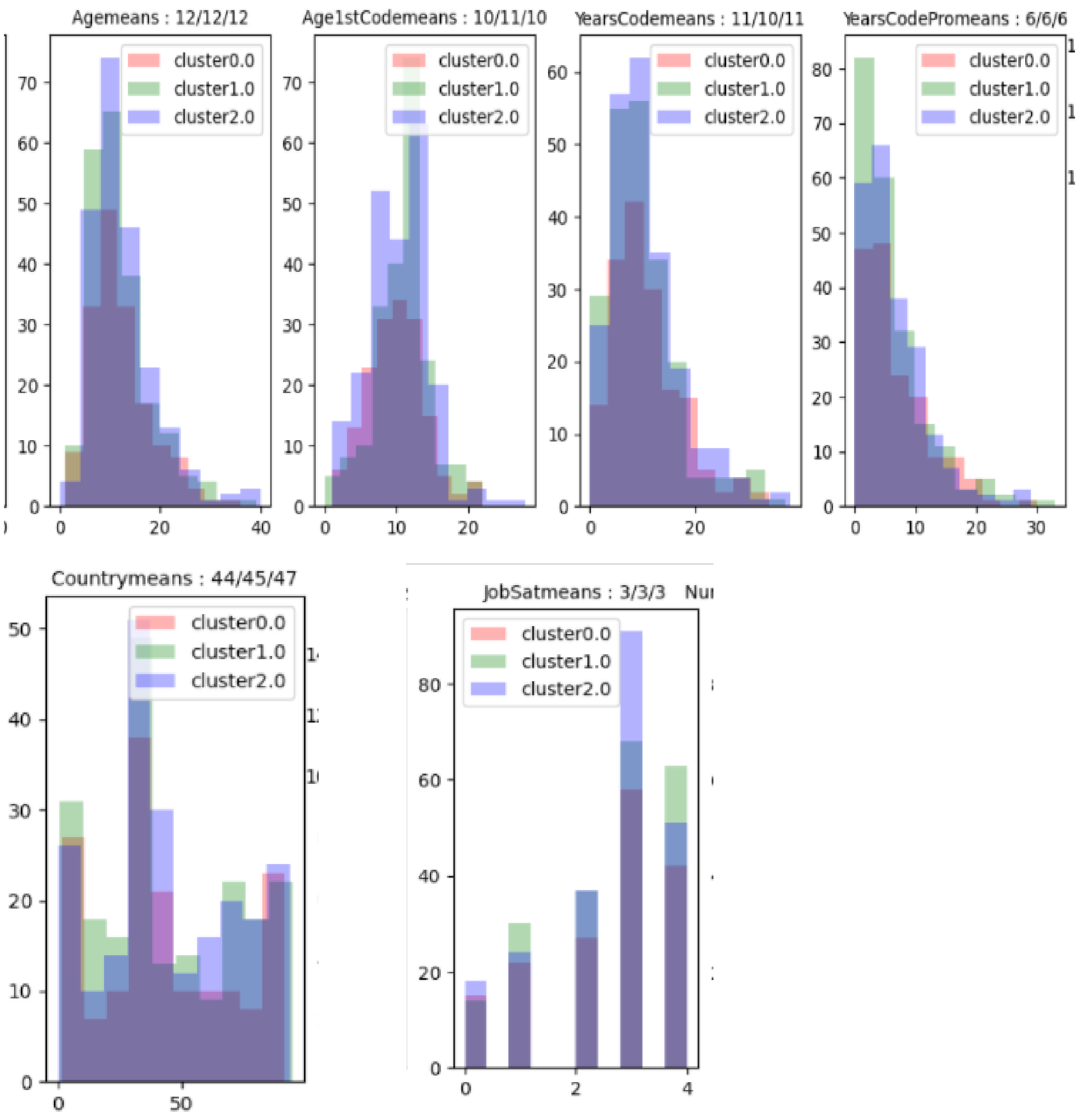
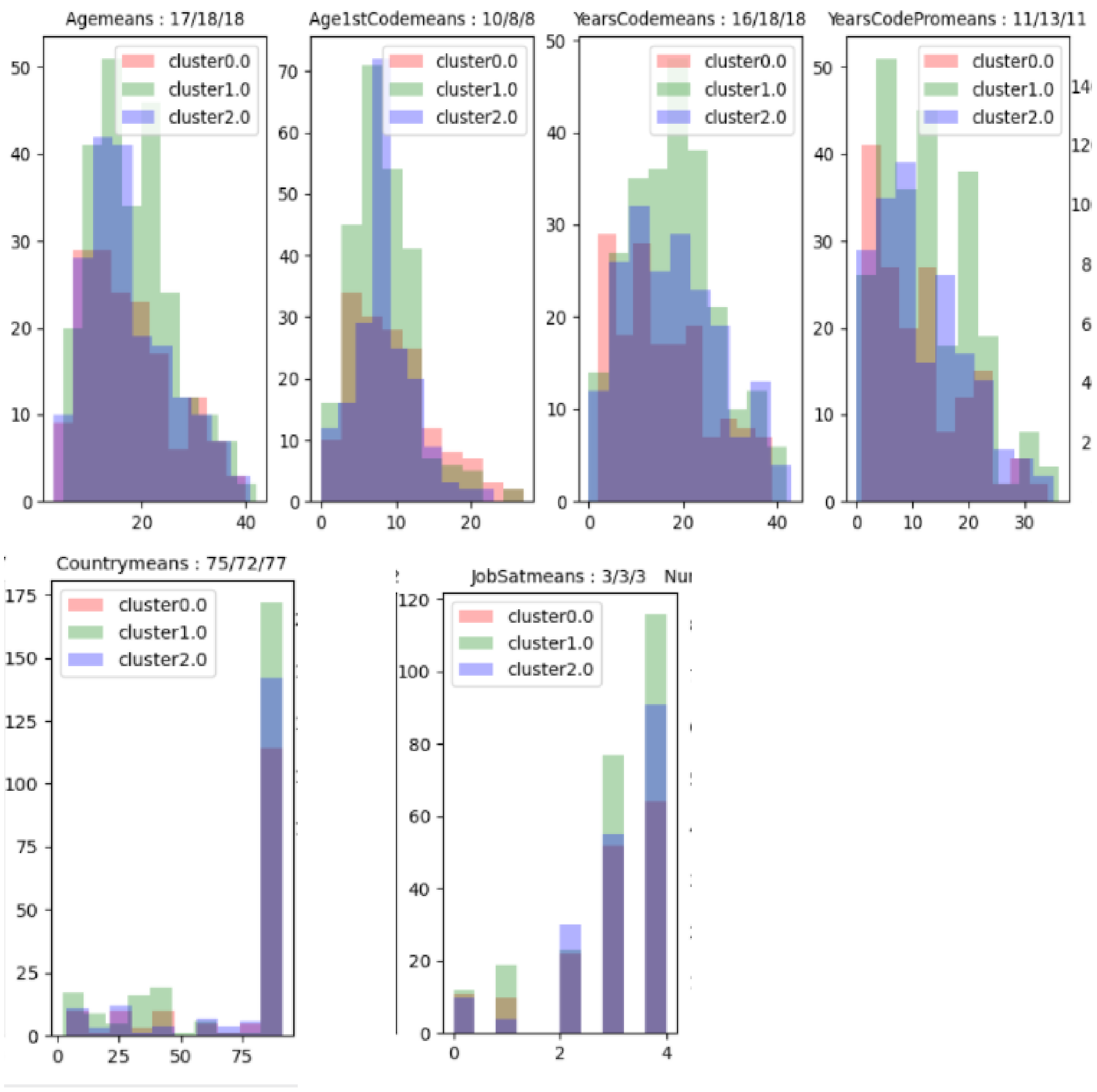


Figure14: High income - Visualizing data with cluster1~3 & each columns



4. Machine Learning for classification and their implementation

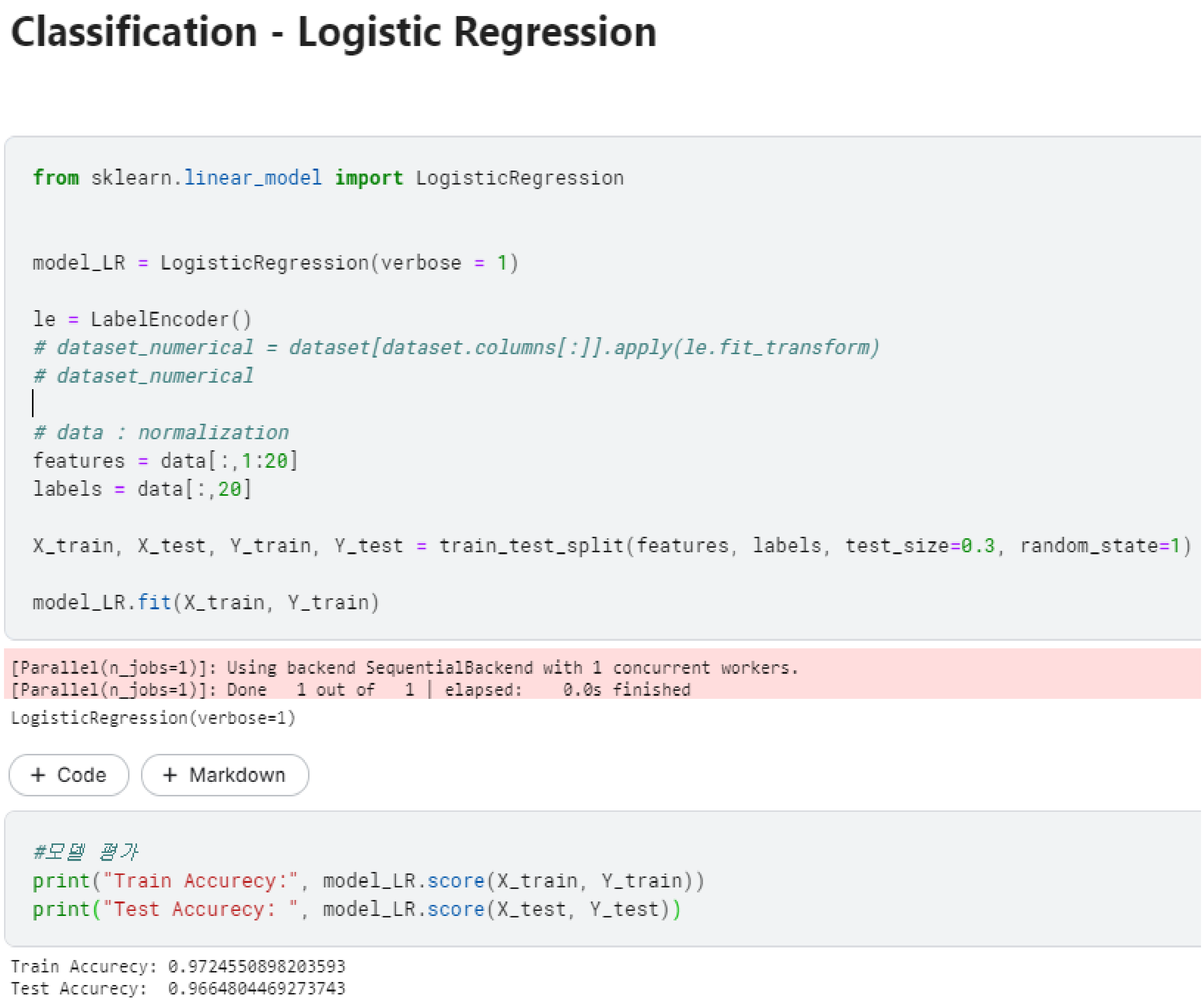
Describe the workflow of machine learning for classification using a flow-chart

In this project, three classifiers will be used, which are “**Logistic Regression**”, “**Artificial Neural networks**” and “**SVM** (support vector machine)”.

First, Logistic Regression is a statistical method for predicting binary classes. It is easy to implement and can be used as the baseline for any binary classification problem. It describes and estimates the relationship between one dependent binary variable and independent variables. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

The processes how to do Logistic Regression is something like following. 1) Loading Data 2) Selecting Feature: it needs to divide the given columns into two types of variable dependent (target variable) and independent variable (feature variables). 3) Splitting Data: divide the dataset into a training set and a test set. 4) Import the Logistic Regression module and create a Logistic Regression classifier object: using **LogisticRegression()function**. 5) Fit the model on the train set: using **fit()** and perform prediction on the test set : using **predict()**.

The Advantages of Logistic Regression are easy to implement, easily interpretable. Also, it does not require scaling of features. The disadvantages of this are not able to handle a large number of categorical features/ variables. (Navlani, December 17, 2019)



Classification - Logistic Regression

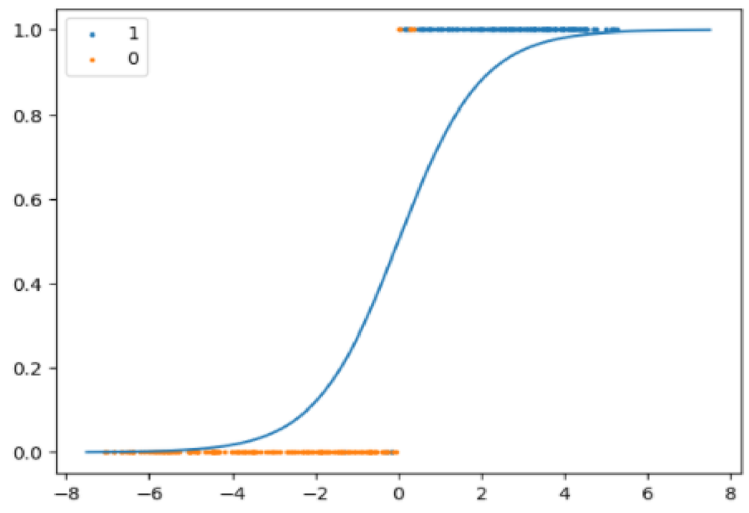
```python
from sklearn.linear_model import LogisticRegression


model_LR = LogisticRegression(verbose = 1)

le = LabelEncoder()
# dataset_numerical = dataset[dataset.columns[:]].apply(le.fit_transform)
# dataset_numerical

# data : normalization
features = data[:,1:20]
labels = data[:,20]

X_train, X_test, Y_train, Y_test = train_test_split(features, labels, test_size=0.3, random_state=1)

model_LR.fit(X_train, Y_train)
```

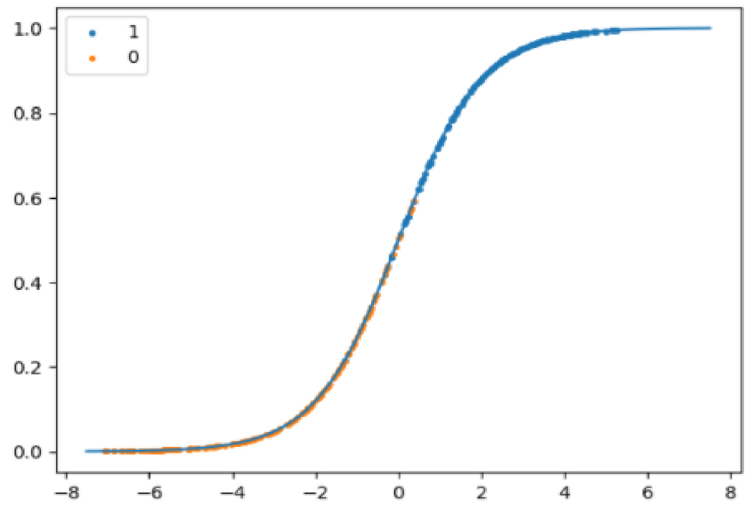
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s finished
LogisticRegression(verbose=1)
```

```python
#모델 평가
print("Train Accurecy:", model_LR.score(X_train, Y_train))
print("Test Accurecy: ", model_LR.score(X_test, Y_test))
```

```
Train Accurecy: 0.9724550898203593
Test Accurecy:  0.9664804469273743
```

```
# plot sigmoid

def sigmoid(x):
  a = []
  for itr in x:
    a.append(1/(1+np.exp(-itr)))

  return a

xs = np.linspace(-7.5,7.5,200)
sig = sigmoid(xs.tolist())

# prediction scatter plot
id_1 = np.where(Y_test == 1)[0].tolist()
id_0 = np.where(Y_test == 0)[0].tolist()


import math
predict = model_LR.predict(X_test)
prob = model_LR.predict_proba(X_test)

ids = np.argmax(prob, 1)

probs = []
for i in range(ids.shape[0]):
  p = prob[i,ids[i]]

  if ids[i] == 0:
    p = 1-p
  probs.append(p)
print(probs)

x_values = []
for i in range(ids.shape[0]):
  x = -math.log((1-probs[i])/probs[i])
  x_values.append(x)



x_values1, predict1, probs1 = np.array(x_values)[id_1], predict[id_1], np.array(probs)[id_1]
x_values0, predict0, probs0 = np.array(x_values)[id_0], predict[id_0], np.array(probs)[id_0]


plt.plot(xs, sig)
# plt.scatter(x_values, predict.tolist())
plt.scatter(x_values1, predict1, label = '1', s = 3)
plt.scatter(x_values0, predict0, label = '0', s = 3)
plt.legend()
plt.show()

plt.plot(xs, sig)
# plt.scatter(x_values, predict.tolist())
plt.scatter(x_values1, probs1, label = '1', s = 6)
plt.scatter(x_values0, probs0, label = '0', s = 6)
plt.legend()
plt.show()
```

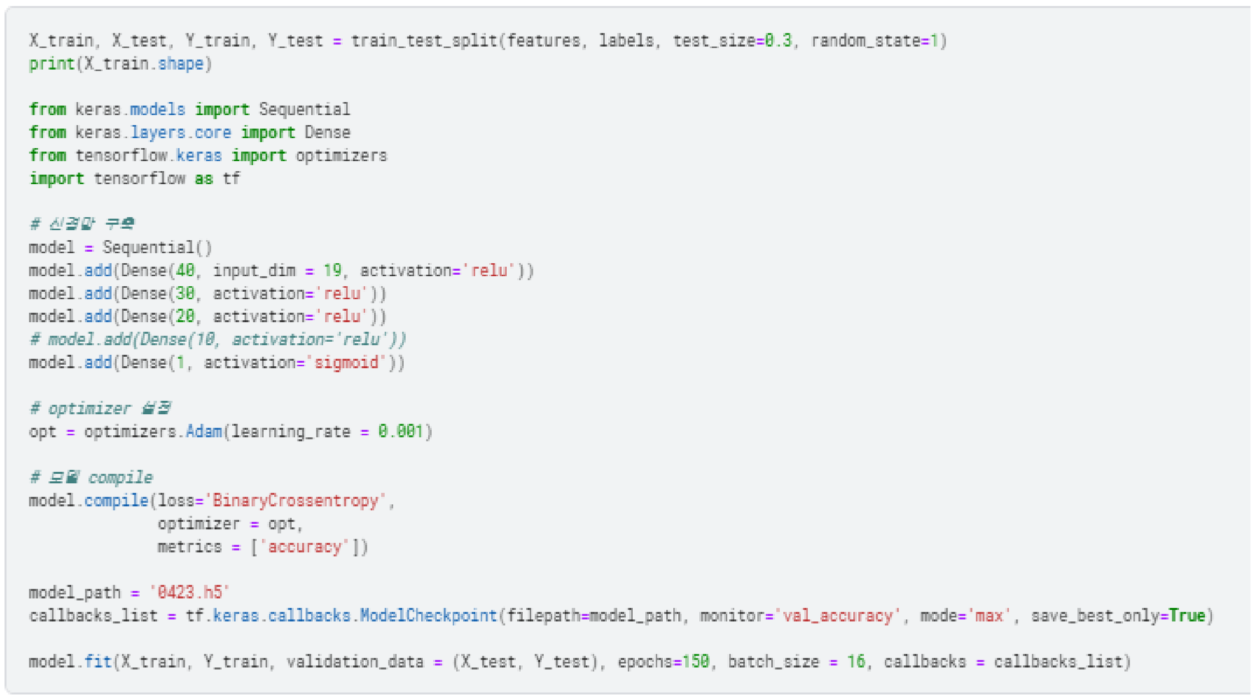




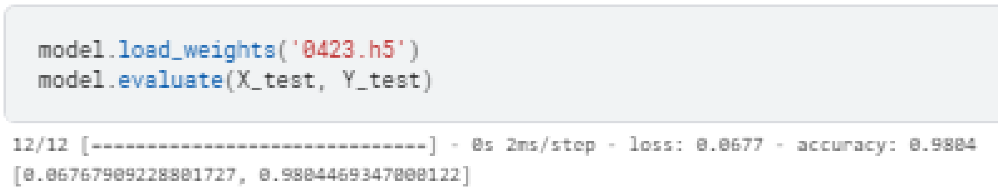
Second, Artificial Neural Networks is a system that learns how to make predictions. ANN architecture is based on the structure and function of the biological neural network. Similar to neurons in the brain, ANN also consists of neurons which are arranged in various layers. Neural networks take input data, train themselves to recognize patterns found in the data, and then predict the output for a new set of similar data. The good thing about Artificial Neural Networks is that same computations can extract information from any kind of data. This means that it does not matter if you are using image data or text data. (Pratik Shukla, Jun 20, 2020)

The advantages of ANN methods are that they require less formal statistical training, ability to implicitly detect complex nonlinear relationships between dependent and independent variables, ability to detect all possible interactions between predictor variables. It is quite robust to noise in the training data. Even the training example may contain errors, which do not affect the final output. It is used the fast evaluation of the learned target function required. Also, ANNs are represented by attribute-value pairs. On the other hand, the disadvantages of ANN methods are it require processor with parallel processing power. So, the realization of equipment is dependent. Also, it does not give a clue as to why and how when ANN gives a probing solution. (Anon., Aug 22 ,2020)

Artificial Neural Network



```
X_train, X_test, Y_train, Y_test = train_test_split(features, labels, test_size=0.3, random_state=1)
print(X_train.shape)

from keras.models import Sequential
from keras.layers.core import Dense
from tensorflow.keras import optimizers
import tensorflow as tf

# 신경망 구축
model = Sequential()
model.add(Dense(40, input_dim = 19, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(20, activation='relu'))
# model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# optimizer 설정
opt = optimizers.Adam(learning_rate = 0.001)

# 모델 compile
model.compile(loss='BinaryCrossentropy',
              optimizer = opt,
              metrics = ['accuracy'])

model_path = '0423.h5'
callbacks_list = tf.keras.callbacks.ModelCheckpoint(filepath=model_path, monitor='val_accuracy', mode='max', save_best_only=True)

model.fit(X_train, Y_train, validation_data = (X_test, Y_test), epochs=150, batch_size = 16, callbacks = callbacks_list)
```



```
model.load_weights('0423.h5')
model.evaluate(X_test, Y_test)
```

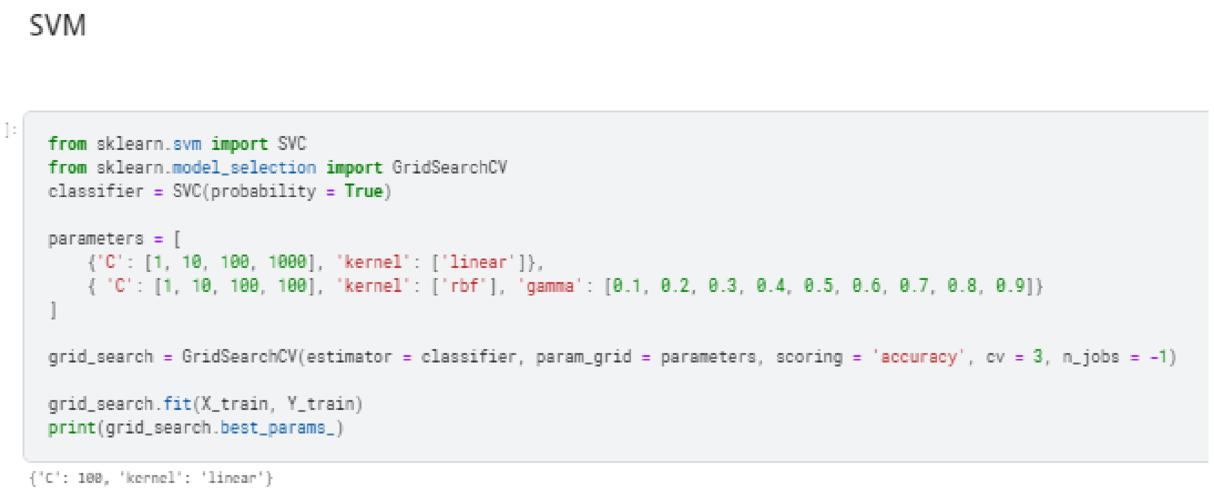
```
12/12 [==============================] - 0s 2ms/step - loss: 0.0677 - accuracy: 0.9804
[0.06767909228801727, 0.9804469347000122]
```

Third, SVM (Support Vector Machine) method. SVM is to find a hyperplane in a N-dimensional space that distinctly classifies the data points. Hyperplanes are decision boundaries that help classify the data points. It separates between a set of objects having different class memberships. Support Vectors are the data points, which are closest to the hyperplane.

How does SVM work? 1) Generate hyperplanes which segregates the classes in the best way. 2) Select the right hyperplane with the maximum segregation from the either nearest data points. The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. 3) Loading Data 4) Exploring Data 5) Splitting Data: dividing the dataset into training and using the function **train_test_split()** 6)Generating Model : import SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function, after then, fit this model on train set using **fit().** Perform prediction on the test set using **predict().**

The advantages of SVM are they offer good accuracy and perform faster and use less memory. The disadvantages of SVM are they are not suitable for large dataset.

SVM is a supervised machine learning algorithm which can be used for both classification or regression challenges. (Navlani, December 28, 2019)



SVM

```
]:
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
classifier = SVC(probability = True)

parameters = [
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    { 'C': [1, 10, 100, 100], 'kernel': ['rbf'], 'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]}
]

grid_search = GridSearchCV(estimator = classifier, param_grid = parameters, scoring = 'accuracy', cv = 3, n_jobs = -1)

grid_search.fit(X_train, Y_train)
print(grid_search.best_params_)
```

```
{'C': 100, 'kernel': 'linear'}
```

To describe data transformation, I have created a table to demonstrate data transformation easily.
The data transformation table is shown below:

1	Initial Data shape	(64461 rows, 61 columns)
2	After Drop Null values, data shape	(1223, 61)
3	Select some features/ columns that seems like related with Salary.	Columns = ['Respondent','MainBranch', 'Hobbyist', 'Gender', 'Age', 'Age1stCode', 'YearsCode', 'YearsCodePro','DevType', 'EdLevel', 'UndergradMajor', 'ConvertedComp', 'WorkWeekHrs', 'Country', 'Ethnicity', 'LanguageDesireNextYear', 'LanguageWorkedWith', 'NEWLearn', 'NEWOvertime', 'JobSat'] Data shape = (1223, 20)
4	Replace all unusual attributes to NaN values, and drop the NaN values	Data shape = (1193, 20)
5	Add ‘incomes’ columns (1 for high income, 0 for low income)	Data shape = (1193, 21)
6	Create new columns ‘NumLanguageWorkedWith’ from ‘LanguageWorkedWith’ ,and replace the ‘LanguageWorkedWith’ columns to ‘NumLanguageWorkedWith’ columns.	Data shape = (1193, 21)

For data normalization, it uses the method, which divides all columns with Max values, so that it can make all data values to 0~1 range (mapping). Figure15 demonstrates how I undertook the normalization process for this project.

Figure15: Normalization

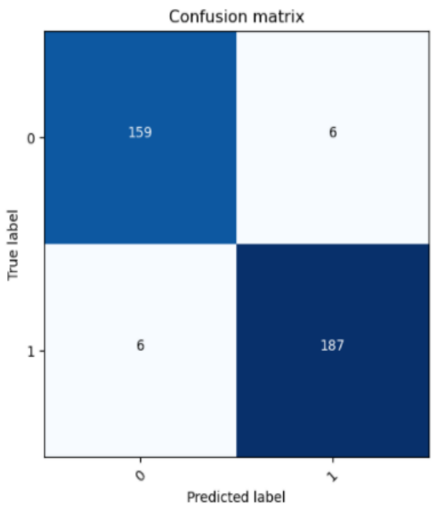
```python
from sklearn.preprocessing import LabelEncoder
import copy
# Change the values to all numerical value
le = LabelEncoder()
encoded_Series = dataset[dataset.columns[:]].apply(le.fit_transform)

data = copy.deepcopy(encoded_Series)
data_origin = copy.deepcopy(dataset.values)

# Normalization of each column
for i in range(20):
  column = data.columns[i]
  data[column] = data[column]/data[column].max()

data = data.values
```

5. Evaluation Machine Learning Models

To evaluate and compare the performance of the models, I make a summary of the performance comparison table.

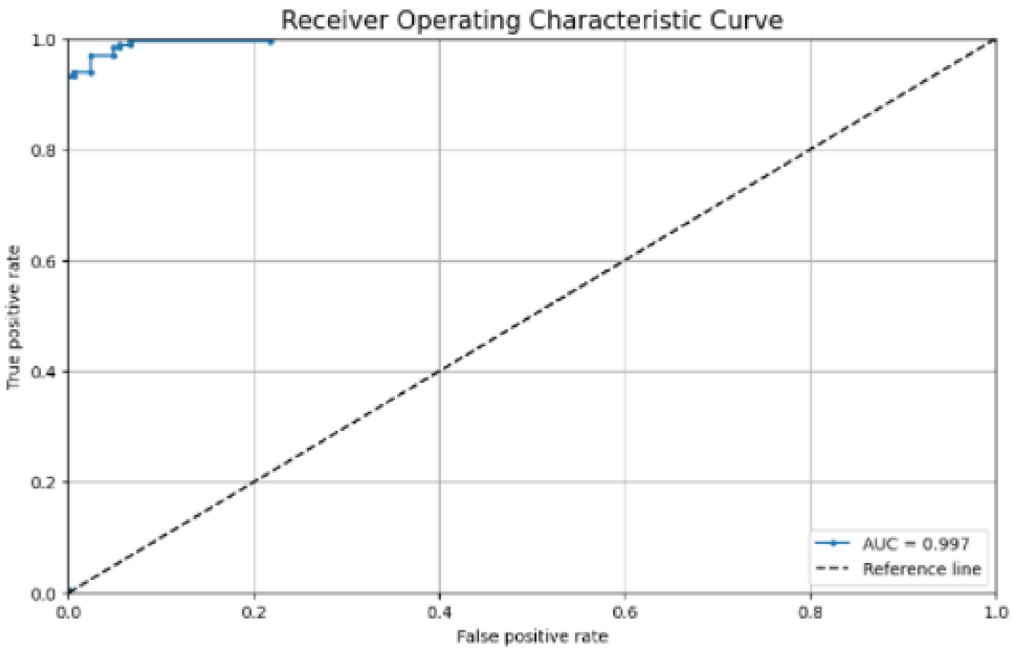
Logistic Regression	

Train-test Accuracy	`#모델 평가`
	`print("Train Accurecy:", model_LR.score(X_train, Y_train))`
	`print("Test Accurecy: ", model_LR.score(X_test, Y_test))`
	
	`Train Accurecy: 0.9724550898203593`
	`Test Accurecy:  0.9664804469273743`

Confusion Matrix	`from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score,roc_curve, auc`
	
	`Y_pred = model_LR.predict(X_test)`
	
	`confusion_mtx = confusion_matrix(Y_test, Y_pred)`
	`plot_confusion_matrix(confusion_mtx, classes = range(2))`
	`# print(classification_report(Y_test,Y_pred))`
	
	`confusion_mtx`
	`sensitivity = confusion_mtx[1,1] / (confusion_mtx[1,1] + confusion_mtx[0,1])`
	`specificity = confusion_mtx[0,0] / (confusion_mtx[0,0] + confusion_mtx[1,0])`
	
	`print('accuracy : {:.4f} / sensitivity : {:.4f} / specificity : {:.4f}'.format(model_LR.score(X_test, Y_test), sensitivity, specificity))`
	
	`accuracy : 0.9665 / sensitivity : 0.9689 / specificity : 0.9636`
	

ROC curve

```
Y_probs = model_LR.predict_proba(X_test)
# print(Y_probs)

fpr , tpr , thresholds = roc_curve (Y_test, Y_probs[:,1])

auc_keras = auc(fpr, tpr)
fig=plt.figure(figsize=(10, 6))
plt.rc('legend', fontsize=10)
plt.plot(fpr, tpr,   marker = '.', label='AUC = {:.3f}'.format(auc_keras))
plt.plot([0, 1], [0, 1], 'k--',label='Reference line')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('False positive rate',fontsize=10)
plt.ylabel('True positive rate',fontsize=10)
plt.title('Receiver Operating Characteristic Curve',fontsize=15)
plt.legend(loc='best')
plt.grid()
plt.show()
```



Artificial Neural Networks

Train-test Accuracy

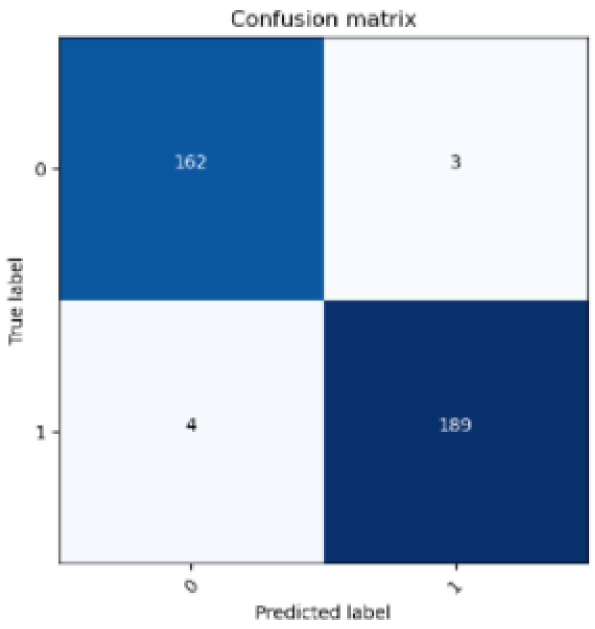
```
model.load_weights('0423.h5')
model.evaluate(X_test, Y_test)
```

```
12/12 [==============================] - 0s 2ms/step - loss: 0.0677 - accuracy: 0.9804
[0.06767909228801727, 0.9804469347000122]
```

Confusion Matrix

```python
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score,roc_curve, auc


Y_pred = model.predict(X_test)
Y_pred = np.round(Y_pred).squeeze()

confusion_mtx = confusion_matrix(Y_test, Y_pred)
plot_confusion_matrix(confusion_mtx, classes = range(2))
print(classification_report(Y_test,Y_pred))
```

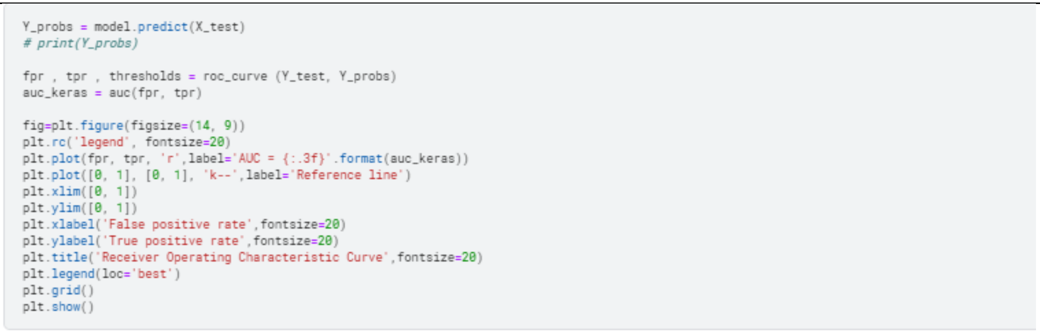
```
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98       165
         1.0       0.98      0.98      0.98       193

    accuracy                           0.98       358
   macro avg       0.98      0.98      0.98       358
weighted avg       0.98      0.98      0.98       358
```



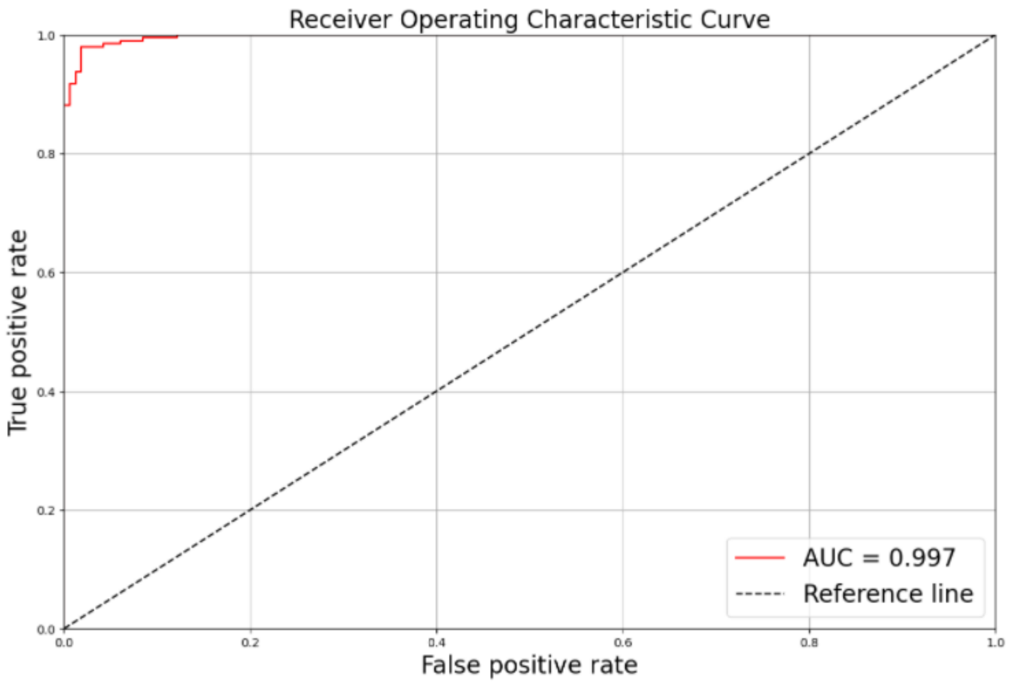
```python
confusion_mtx
sensitivity = confusion_mtx[1,1] / (confusion_mtx[1,1] + confusion_mtx[0,1])
specificity = confusion_mtx[0,0] / (confusion_mtx[0,0] + confusion_mtx[1,0])

print('sensitivity : {:.4f} / specificity : {:.4f}'.format(sensitivity, specificity))
```

```
sensitivity : 0.9844 / specificity : 0.9759
```

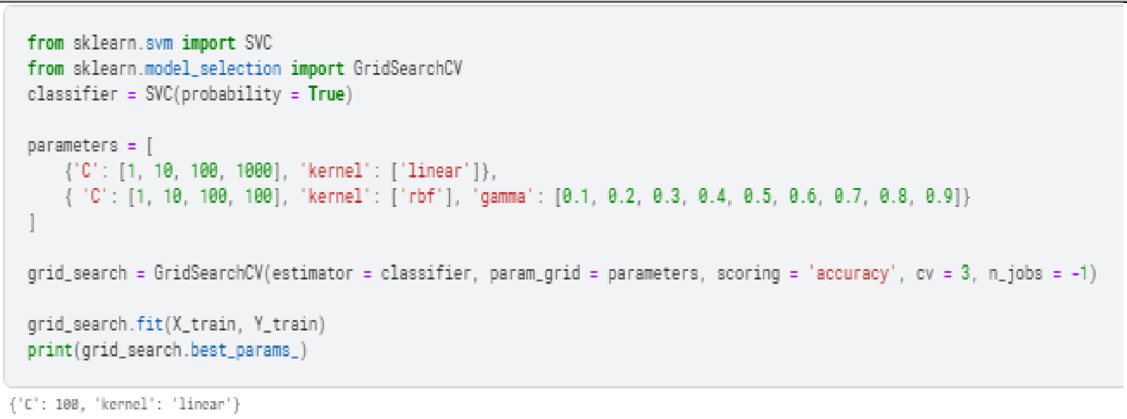
ROC curve



```
Y_probs = model.predict(X_test)
# print(Y_probs)

fpr , tpr , thresholds = roc_curve (Y_test, Y_probs)
auc_keras = auc(fpr, tpr)

fig=plt.figure(figsize=(14, 9))
plt.rc('legend', fontsize=20)
plt.plot(fpr, tpr, 'r',label='AUC = {:.3f}'.format(auc_keras))
plt.plot([0, 1], [0, 1], 'k--',label='Reference line')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('False positive rate',fontsize=20)
plt.ylabel('True positive rate',fontsize=20)
plt.title('Receiver Operating Characteristic Curve',fontsize=20)
plt.legend(loc='best')
plt.grid()
plt.show()
```



SVM (Support Vector Machine)



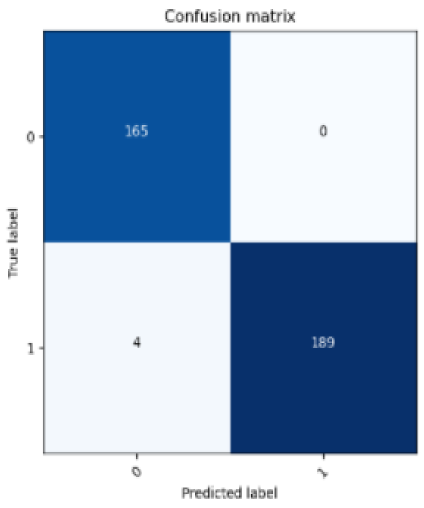
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
classifier = SVC(probability = True)

parameters = [
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    { 'C': [1, 10, 100, 100], 'kernel': ['rbf'], 'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]}
]

grid_search = GridSearchCV(estimator = classifier, param_grid = parameters, scoring = 'accuracy', cv = 3, n_jobs = -1)

grid_search.fit(X_train, Y_train)
print(grid_search.best_params_)
```

```
{'C': 100, 'kernel': 'linear'}
```

Confusion Matrix

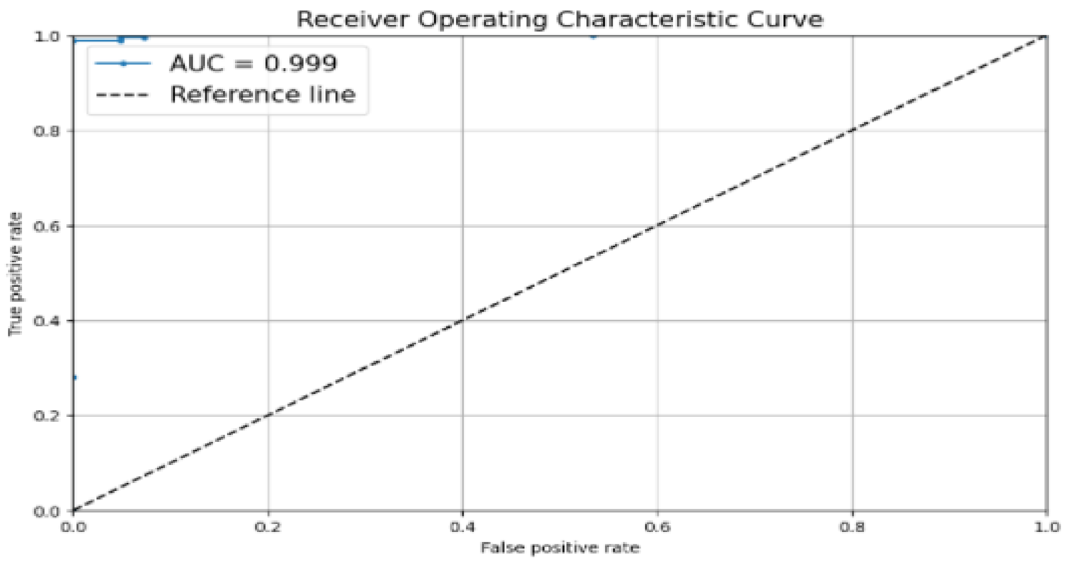
```python
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score,roc_curve, auc


Y_pred = grid_search.predict(X_test)

confusion_mtx = confusion_matrix(Y_test, Y_pred)
plot_confusion_matrix(confusion_mtx, classes = range(2))
# print(classification_report(Y_test,Y_pred))

confusion_mtx
sensitivity = confusion_mtx[1,1] / (confusion_mtx[1,1] + confusion_mtx[0,1])
specificity = confusion_mtx[0,0] / (confusion_mtx[0,0] + confusion_mtx[1,0])

print('accuracy : {:.4f} / sensitivity : {:.4f} / specificity : {:.4f}'.format(grid_search.score(X_test, Y_test), sensitivity, specificity))
```

```
accuracy : 0.9888 / sensitivity : 1.0000 / specificity : 0.9763
```



ROC curve

```python
Y_probs = grid_search.predict_proba(X_test)
# print(Y_probs)

fpr , tpr , thresholds = roc_curve (Y_test, Y_probs[:,1])

auc_keras = auc(fpr, tpr)
fig=plt.figure(figsize=(10, 6))
plt.rc('legend', fontsize=15)
plt.plot(fpr, tpr,   marker = '.', label='AUC = {:.3f}'.format(auc_keras))
plt.plot([0, 1], [0, 1], 'k--',label='Reference line')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('False positive rate',fontsize=10)
plt.ylabel('True positive rate',fontsize=10)
plt.title('Receiver Operating Characteristic Curve',fontsize=15)
plt.legend(loc='best')
plt.grid()
plt.show()
```



Summary of each classification methods

	Logistic Regression	Artificial Neural Network	SVM (Support Vector Machine)
Accuracy	Train Accuracy: 0.9724 → 97.24 % Test Accuracy: 0.9665 →96.65 %	Accuracy: 0.9804 → 98.04 % Loss → 0.0677	Accuracy: 0.9888 → 98.88 %
Confusion Matrix	***Sensitivity:*** TP/Actual yes 187/ (187+6) = 0.9689 →96.89 % ***Specificity:*** TN/Actual no 159/ (159+6)= 0.9636 →96.36 %	***Sensitivity:*** TP/Actual yes 189/ (189 + 4) = 0.9792 →97.92 % ***Specificity:*** TN/Actual no 162/ (162+3) = 0.9818 →98.18 %	***Sensitivity:*** TP/Actual yes 189/ (189 + 4) = 0.9792 →97.92 % ***Specificity:*** TN/Actual no 165/ (165 + 0) = 1.0 →100 %
ROC curve	AUC = 0.997	AUC = 0.997	AUC = 0.999

Explanation for each term

<Confusion matrix>

	Prediction		
Actual Class		1 (Positive)	0 (Negative)
	1 (Positive)	True Positive (TP)	False Negative (FN)
	0 (Negative)	False Positive (FP)	True Negative(TN)

1) True Positives (TP): Predicted yes(they are high income), and they actually have high income.
2) True Negatives (TN): Predicted no, and they are not high income
3) False Positives (FP): predicted yes, but they are not high income (Type1 error)
4) False Negatives (FN): Predicted no, but they actually have high income. (Type2 error)

<Accuracy>

Overall, how often is the classifier correct?

→ (TP + TN)/ Total

<True Positive Rate (= Sensitivity or Recall)>

When it is actually yes, how often does it predict yes?

→ TP/Actual yes

<True Negative Rate (= Specificity)>

When it is actually no, how often does it predict no?

→ TN/Actual no

<Precision>

When it predicts yes, how often is it correct?

→ TP/Predicted yes

<What is the AUC-ROC curve?>

ROC (Reciever Operator Characteristic) curve is an evaluation metric for binary classification problems. ROC is a probability curve for different classes. It tells us how good the model is for distinguishing the given classes, in terms of the predicted probability. A typical ROC curve has **False Positive Rate(FPR)** on X-axis and **True Positive Rate (TPR)** on the Y-axis. The AUC (Area Under Curve) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of ROC curve. The **higher AUC, the better the performance of the model** at distinguishing between the positive and negative classes.

Explanation for each classification methods

1) Logistic Regression
2) Artificial Neural Network: It uses four layers, and node number is 40/30/20. Also, it uses early-stopping methods. For hyper-parameters, it uses **Adam, epochs**=150, **batch_size**=32. Also it uses 'BinaryCrossentropy' **loss**.
3) SVM (Support Vector Machine): It uses 'linear' and 'rbf' **kernel** and **gamma**[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. Also, it uses **GrindSearchCV.**

As the result, SVM (Support Vector Machine) shows the best performance. It has the highest accuracy as 98.88% and it has high AUC = 0.999. It means SVM has the better performance than the other classification methods (Logistic Regression, Artificial Neural Network). SVM is the most suitable classification methods on this project dataset. I think it is because this project dataset is not too large, so it makes the good accuracy and best performance by using SVM methods.

6. Discussions and Conclusions

To summarize this coursework, this coursework aimed to find which features were the most relevant in causing developers to earn a high income. In this dataset, nearly 65,000 developers participated in the survey, and they answered questions about how they learn and level up, which tools they are using and what they want. Most of our survey respondents in 2020 were professional developers, people who code sometimes as part of their work or students preparing for such a career. To analyze about developers' roles, about 55% of respondents identify as full-stack developers, and about 20% are considered as mobile developers. Also, to compare by gender, there are small increases in the representation of developer roles that have the most representation from women, like Data Scientists and Academic Researchers. About 78% of respondents say that they code as a hobby. In regard to experience, there are quite a sizable percentage of developers who learned to code more than 30 years ago (around 15%). There are around 40% of developers who learned to code less than 1 year ago. Concerning education, around 75% of respondents worldwide completed at least a bachelor's degree or higher. They mostly hold a degree in computer science, computer engineering, or software engineering. For career values, almost 65% respondents said that they are either slightly or very satisfied with their job.

The first insight that I have gained from this coursework is that I got quite high results of accuracy from classification methods. From this, I can conclude that depends it on the complexity of tasks and that the size of dataset did not affect the result as much. The raw dataset has lots of data but after data processing, the dataset becomes smaller. However, it seems like it did not affect to the result that much because this dataset complexity was not too complicated. Secondly, I have checked the results before and after normalization. After doing normalization, more accurate data results are generated. Therefore, I found that normalization is an important process when people are doing data analytics.

The overall result is that SVM classification methods were the best methods in this dataset. I think it is because the dataset was 1D data, and the number of selected features were small. I think that SVM classification methods are suitable when the dataset is not too big and not complicated.

References

[1]Anon., Aug 22 ,2020. *Asquero.* [Online]
Available at: https://www.asquero.com/article/advantages-and-disadvantages-of-artificial-neural-networks/
[Accessed 27 April 2021].

automaticaddison, July 23, 2019. *automatic addison.* [Online]
Available at: https://automaticaddison.com/advantages-of-k-means-clustering/
[Accessed April 20, 2021].

Dabbura, I., Sep 18, 2018. *towards data science.* [Online]
Available at: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
[Accessed April 20, 2021].

Ersi, N. K., Auguest 1, 2020. *predictive hacks.* [Online]
Available at: https://predictivehacks.com/k-means-elbow-method-code-for-python/#:~:text=K%2DMeans%20is%20an%20unsupervised,optimal%20for%20the%20specific%20case.
[Accessed 23 April 2021].

Kumar, B., March 5,2021. *pythonguides.* [Online]
Available at: https://pythonguides.com/crosstab-in-python-pandas/#:~:text=A%20key%20benefit%20of%20the,returning%20values%20displayed%20as%20percentages.&text=%27all%27%20or%20True%20%E2%80%93%20normalizes,total%20across%20rows%20and%20columns)
[Accessed April 19, ,2021].

Navlani, A., December 17, 2019. *datacamp.* [Online]
Available at: https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=3326020343
[Accessed 24 April 2021].

Navlani, A., December 28, 2019. *datacamp.* [Online]
Available at: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
[Accessed 28 April 2021].

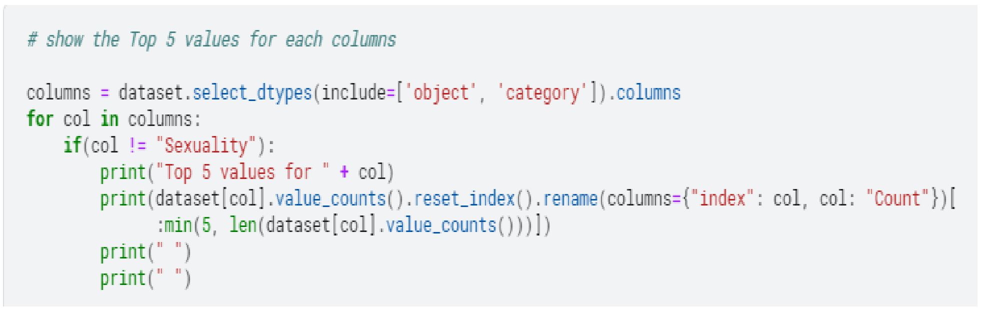
Pratik Shukla, R. I., Jun 20, 2020. *TowardsAi.* [Online]

Available at: https://pub.towardsai.net/building-neural-networks-from-scratch-with-python-code-and-math-in-detail-i-536fae5d7bbf
[Accessed 24 April 2021].

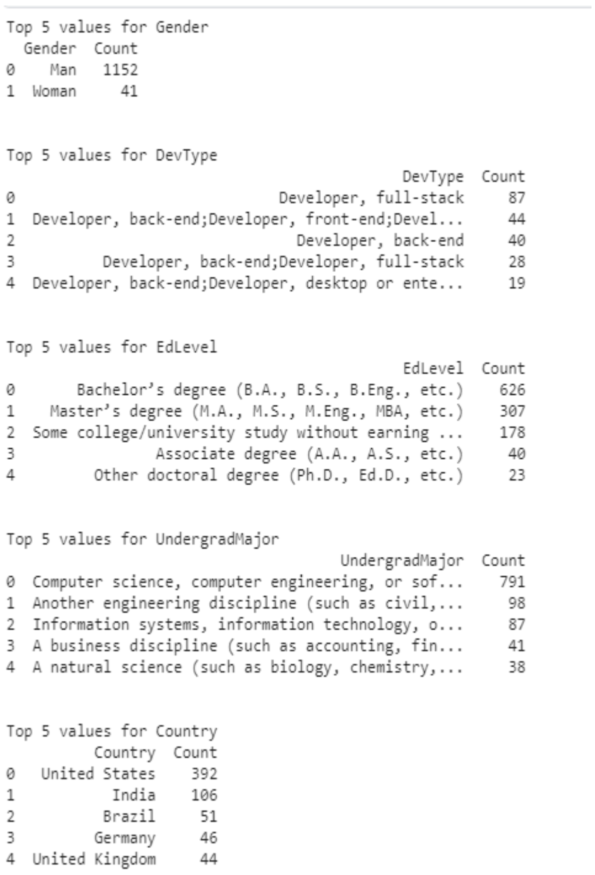
TQP, March 17,2020. *Techqualitypedia.* [Online]
Available at: https://techqualitypedia.com/histogram/
[Accessed April 18, 2021].

Appendix

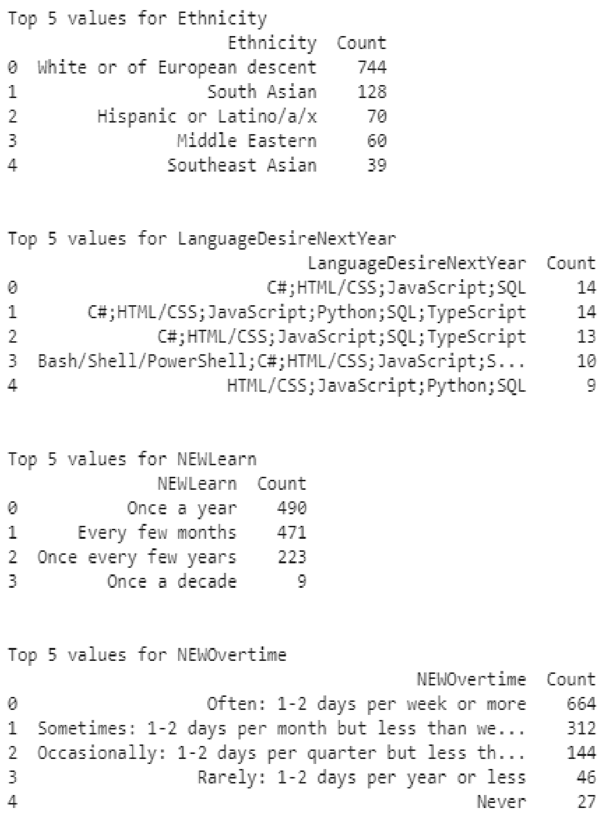
Figure 1



```python
# show the Top 5 values for each columns

columns = dataset.select_dtypes(include=['object', 'category']).columns
for col in columns:
    if(col != "Sexuality"):
        print("Top 5 values for " + col)
        print(dataset[col].value_counts().reset_index().rename(columns={"index": col, col: "Count"})[
            :min(5, len(dataset[col].value_counts()))])
        print(" ")
        print(" ")
```

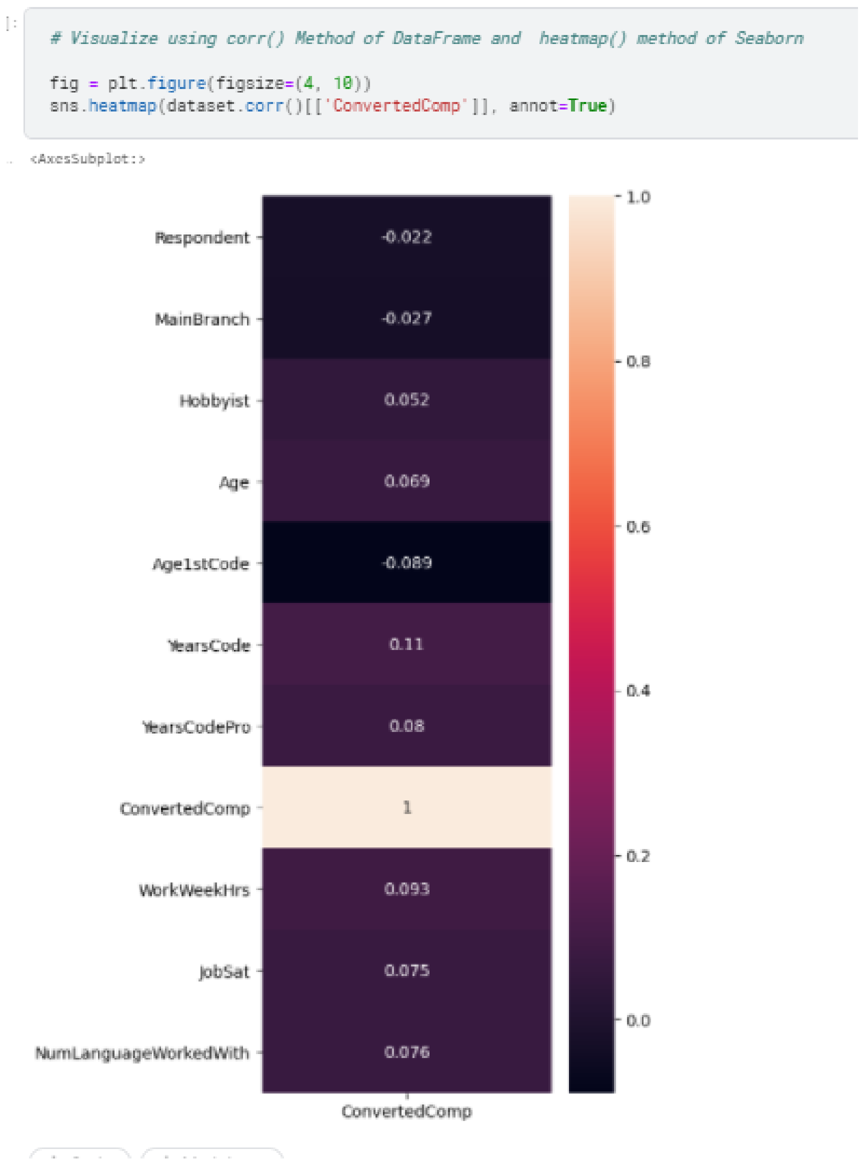


```
Top 5 values for Gender
  Gender  Count
0    Man   1152
1  Woman     41


Top 5 values for DevType
                                             DevType  Count
0                              Developer, full-stack     87
1  Developer, back-end;Developer, front-end;Devel...     44
2                                Developer, back-end     40
3            Developer, back-end;Developer, full-stack     28
4  Developer, back-end;Developer, desktop or ente...     19


Top 5 values for EdLevel
                                             EdLevel  Count
0     Bachelor’s degree (B.A., B.S., B.Eng., etc.)    626
1    Master’s degree (M.A., M.S., M.Eng., MBA, etc.)    307
2  Some college/university study without earning ...    178
3                Associate degree (A.A., A.S., etc.)     40
4        Other doctoral degree (Ph.D., Ed.D., etc.)     23


Top 5 values for UndergradMajor
                                      UndergradMajor  Count
0  Computer science, computer engineering, or sof...    791
1  Another engineering discipline (such as civil,...     98
2  Information systems, information technology, o...     87
3  A business discipline (such as accounting, fin...     41
4  A natural science (such as biology, chemistry,...     38


Top 5 values for Country
          Country  Count
0   United States    392
1           India    106
2          Brazil     51
3         Germany     46
4  United Kingdom     44
```



```
Top 5 values for Ethnicity
                       Ethnicity  Count
0  White or of European descent    744
1                    South Asian    128
2            Hispanic or Latino/a/x     70
3                 Middle Eastern     60
4               Southeast Asian     39


Top 5 values for LanguageDesireNextYear
                              LanguageDesireNextYear  Count
0                         C#;HTML/CSS;JavaScript;SQL     14
1       C#;HTML/CSS;JavaScript;Python;SQL;TypeScript     14
2              C#;HTML/CSS;JavaScript;SQL;TypeScript     13
3  Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;S...     10
4                   HTML/CSS;JavaScript;Python;SQL      9


Top 5 values for NEWLearn
             NEWLearn  Count
0         Once a year    490
1    Every few months    471
2  Once every few years    223
3      Once a decade      9


Top 5 values for NEWOvertime
                                         NEWOvertime  Count
0                      Often: 1-2 days per week or more    664
1  Sometimes: 1-2 days per month but less than we...    312
2  Occasionally: 1-2 days per quarter but less th...    144
3                      Rarely: 1-2 days per year or less     46
4                                               Never     27
```

Figure 2



```
# Visualize using corr() Method of DataFrame and  heatmap() method of Seaborn

fig = plt.figure(figsize=(4, 10))
sns.heatmap(dataset.corr()[['ConvertedComp']], annot=True)
```

```
<AxesSubplot:>
```

Figure 3

EDA - Pie Chart

```
plt.subplots(figsize=(5,5))
data=dataset["Hobbyist"]
data.value_counts().plot.pie(autopct='%1.1f%%',colors=sns.color_palette("dark",5),startangle=90,wedgeprops = { "linewidth" : 2, "edgecolor" : "white"})
plt.title("Do You Code As a Hobby?")
my_circle=plt.Circle((0,0), 0.7, color="white")
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```

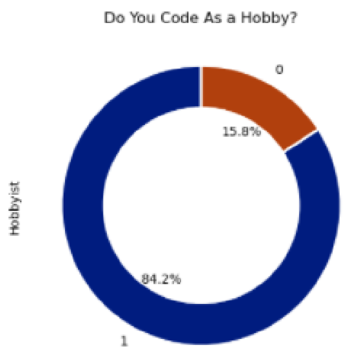


Figure 4

EDA - Barplot

```
plt.figure(figsize=(7,5))
chart = sns.barplot(
    data=dataset,
    x="Gender",
    y="ConvertedComp",
    palette="Set1"
)
chart=chart.set_xticklabels(
    chart.get_xticklabels(),
    rotation=65,
    horizontalalignment="right"
)
```

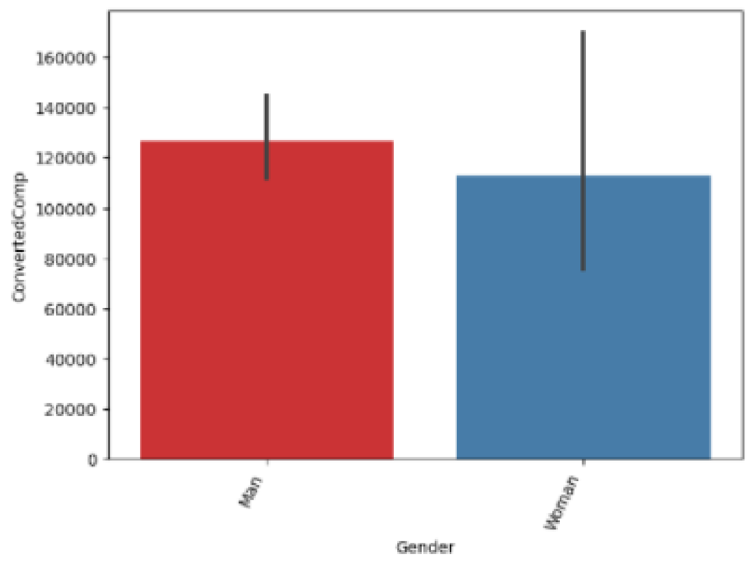


Figure 5

```
df_Ethnicity = dataset.groupby(['Ethnicity']).count().sort_values('Respondent', ascending=False)
races = [str(x) for x in df_Ethnicity.index[0:10]]
pies = [df_Ethnicity.Respondent[df_Ethnicity.index==races[x]][0] for x in range(10)]

fig, ax = plt.subplots(figsize=(10,10))
ax.pie(
    pies,
    labels=races,
    autopct='%.1f%%',
    explode=[0.01, 0.02, 0.03, 0.04, 0.04, 0.04, 0.05, 0.08, 0.1, 0.25],
    startangle=-150, pctdistance=0.7, labeldistance=1.05
)

fig.tight_layout()

plt.show()
plt.style.use('default')
```

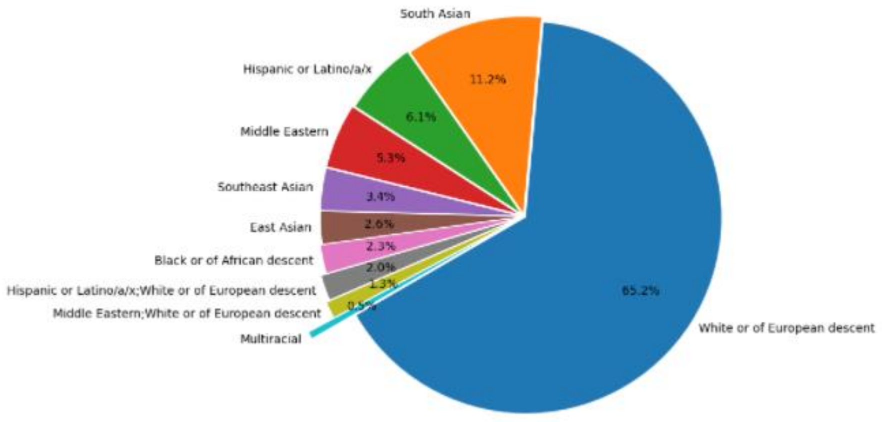


Figure 6

```
#salary=dataset[["ConvertedComp","Gender","Country","MainBranch","UndergradMajor","EdLevel"]].dropna()
sal_job=dataset.groupby("Country")["ConvertedComp"].median().to_frame().sort_values(by="ConvertedComp",ascending=False).head(20)
ax=sns.barplot(sal_job.ConvertedComp,sal_job.index,palette=sns.color_palette("icefire",20))
plt.title("Median Salary by Country",size=15)
for i, v in enumerate(sal_job.ConvertedComp):
    ax.text(.5, i, v,fontsize=10,color="white",weight="bold")
fig=plt.gcf()
fig.set_size_inches(10,10)
plt.show()
```

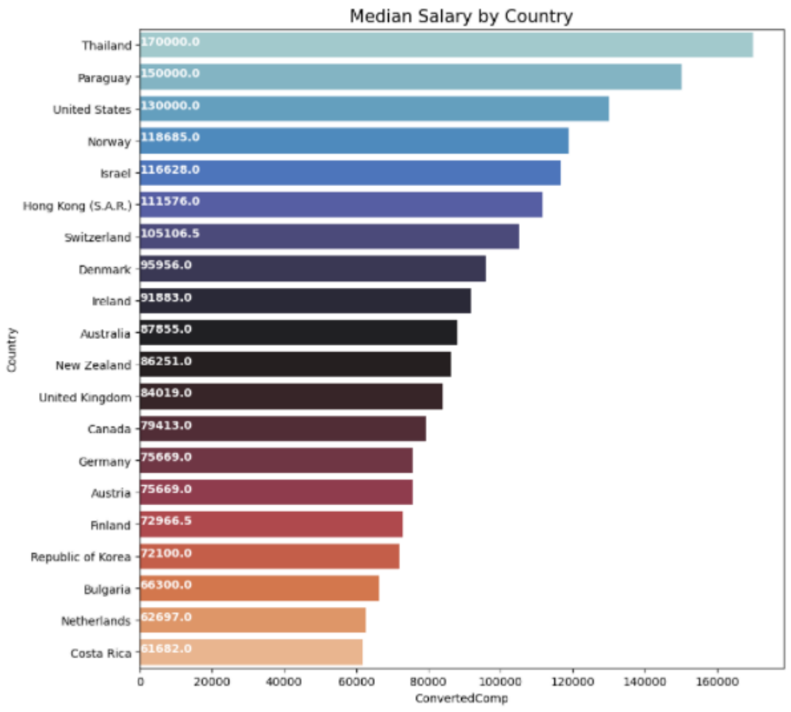


Figure 7

```
sal_job=dataset.groupby("DevType")["ConvertedComp"].median().to_frame().sort_values(by="ConvertedComp",ascending=False).head(20)
ax=sns.barplot(sal_job.ConvertedComp,sal_job.index,palette=sns.color_palette('inferno',20))
plt.title("Median-Salary by Profession",size=15)
plt.ylabel("Developer Role")
for i, v in enumerate(sal_job.ConvertedComp):
    ax.text(.5, i, v,fontsize=10,color='white',weight='bold')
fig=plt.gcf()
fig.set_size_inches(10,10)
plt.show()
```

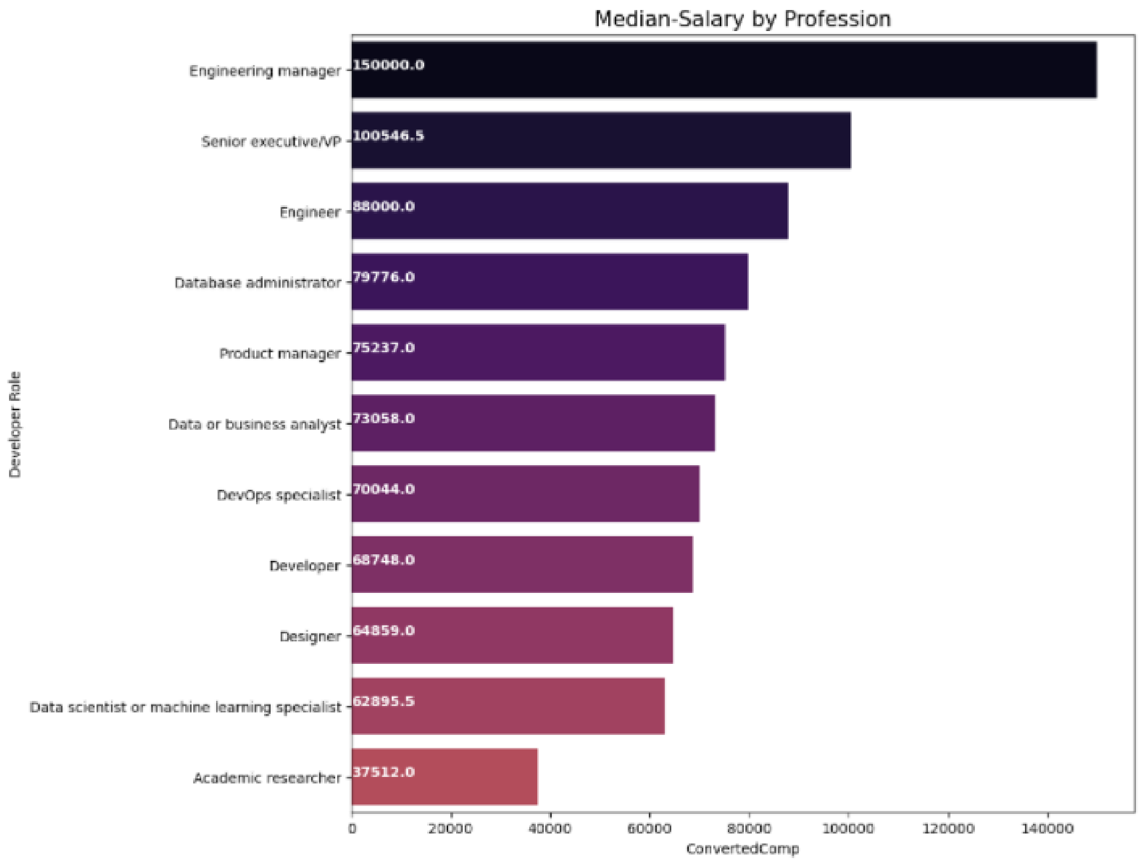


Figure 8

Education Level

```
dataset['EdLevel'].value_counts().plot(kind = 'pie', title='Education Level', autopct='%1.1f%%')
```

```
<AxesSubplot:title={'center':'Education Level'}, ylabel='EdLevel'>
```

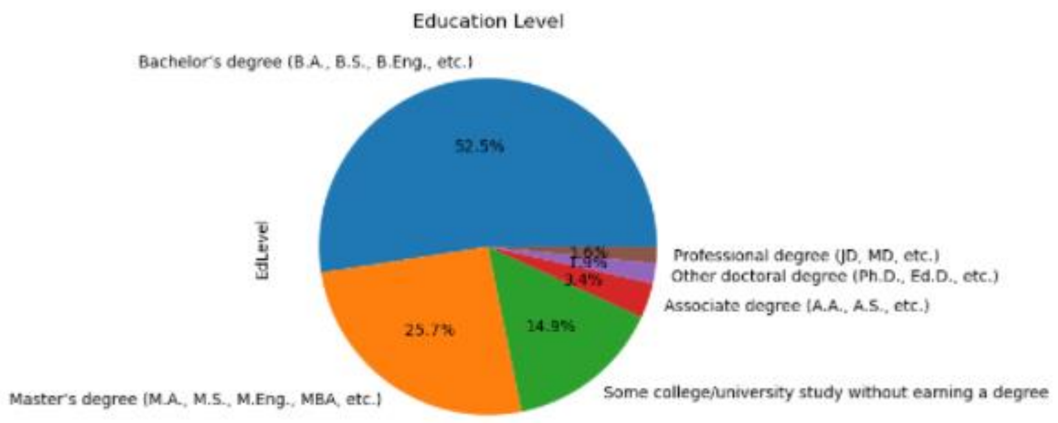
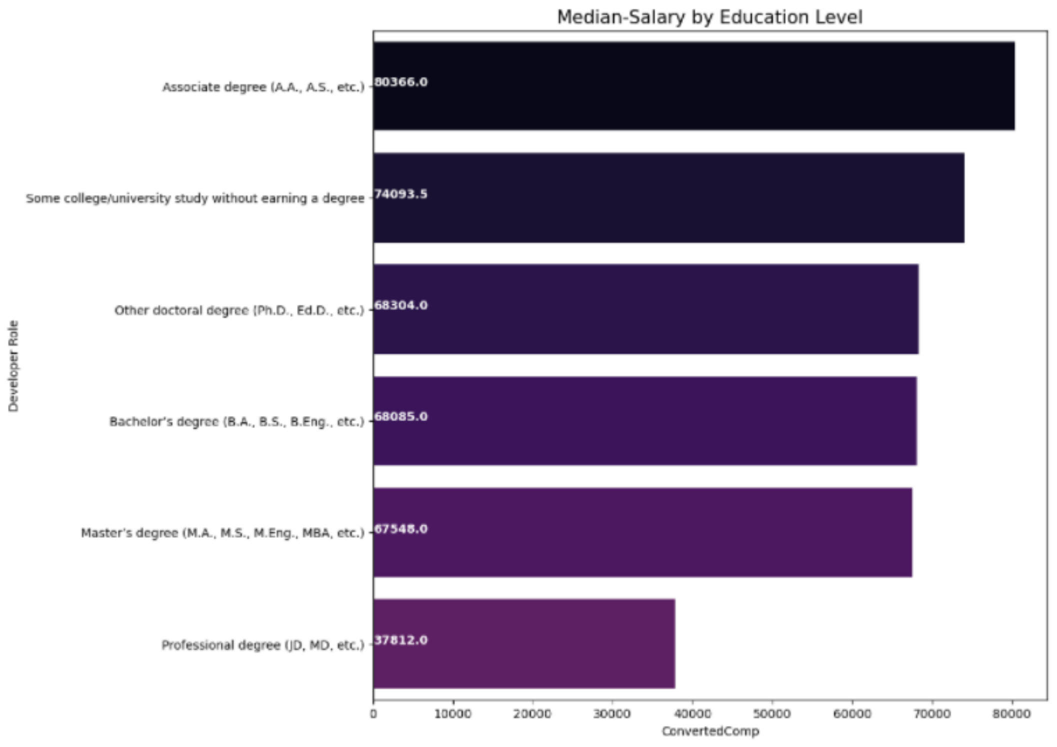


Figure 9

```
sal_ed=dataset.groupby("EdLevel")["ConvertedComp"].median().to_frame().sort_values(by="ConvertedComp",ascending=False).head(20)
ax=sns.barplot(sal_ed.ConvertedComp,sal_ed.index,palette=sns.color_palette("inferno",20))
plt.title("Median-Salary by Education Level",size=15)
plt.ylabel("Developer Role")
for i, v in enumerate(sal_ed.ConvertedComp):
    ax.text(.5, i, v,fontsize=10,color='white',weight='bold')
fig=plt.gcf()
fig.set_size_inches(10,10)
plt.show()
```


Median-Salary by Education Level
Associate degree (A.A., A.S., etc.)
Some college/university study without earning a degree
Other doctoral degree (Ph.D., Ed.D., etc.)
Bachelor's degree (B.A., B.S., B.Eng., etc.)
Master's degree (M.A., M.S., M.Eng., MBA, etc.)
Professional degree (JD, MD, etc.)
Developer Role
ConvertedComp