BA Final Group Project

Group Member: Yunpeng Hou | Yihuan He | Tianzi Zheng | Ziqing Li | Lifu Sun | Jesus Alejandro Garza

Human Resources & Workforce Allocation Analysis

Brief Description:

A HR company has demographic data about people who are looking for a job, and what position they end up having, with what salary and how long does it take for them to get the job

Problem Statements:

Can we predict what might their wages be, what's their position, their education or how long will it take for them to get hired based on the information that we already have?

By solving above problems, we can minimize the resources of HR company spent in matching candidates with the right job and salary.

Project Structure

Data Cleaning: BA

Data Filtering: FT & PT

1. Descriptive Data Mining
   1.1 Data Visualization
   1.2 K-Means

2. Predictive Data Mining

   2.1 Prediction for Numerical Data
   2.1.1 Full-Time DayDiff & Part-Time DayDiff

       2.1.1 a) Linear Regression

   2.1.2 Full-Time TotalWages

       2.1.2 a) Linear Regression
       2.1.2 b) Regression Tree

   2.1.3 Part-Time TotalWages

       2.1.3 a) Linear Regression
       2.1.3 b) Regression Tree

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
BA = pd.read_csv('Employment_Now_Data.csv')
```

In [3]:

```
BA
```

Out[3]:

	PID	AGE	GENDER	ZIP CODE	COUNTY	PROGRAM START DATE	POSITION	JOB START DATE	JOB TERMINATION DATE
0	PID02994	29	male	11692	Arverne	2015/6/24	Administrative	2015/9/3	NaN
1	PID00542	50	male	10456	Bronx	2015/7/4	Administrative	2015/8/3	NaN
2	PID03451	50	male	11420	South Ozone Park	2014/8/29	Administrative	2015/9/8	NaN
3	PID00543	50	male	10456	Bronx	2015/4/25	Administrative	2015/8/3	NaN
4	PID11335	40	male	11220	Brooklyn	2015/3/10	Administrative	2015/8/17	NaN

In [4]:

```
BA.dtypes
```

Out[4]:

```
PID                     object
AGE                      int64
GENDER                  object
ZIP CODE                 int64
COUNTY                  object
PROGRAM START DATE      object
POSITION                object
JOB START DATE          object
JOB TERMINATION DATE    object
WAGE                   float64
HOURS PER WEEK         float64
EDUCATION               object
Quarter and Year        object
dtype: object
```

Data Cleaning

In [5]:

```
BA.drop(["ZIP CODE","COUNTY","JOB TERMINATION DATE"],axis=1,inplace=True)
BA.head()
# There are too many Zip Code and County, so we just dropped
# Because it may cause overfitted problem as we reviewed and assumed there are too many different v
# We will not analyze job termination date.
```

Out[5]:

	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATION
0	PID02994	29	male	2015/6/24	Administrative	2015/9/3	15.60	30.0	HS Graduate
1	PID00542	50	male	2015/7/4	Administrative	2015/8/3	8.75	30.0	HS Graduate
2	PID03451	50	male	2014/8/29	Administrative	2015/9/8	16.00	35.0	HS Graduate
3	PID00543	50	male	2015/4/25	Administrative	2015/8/3	8.75	30.0	HS Graduate
4	PID11335	40	male	2015/3/19	Administrative	2015/8/17	13.92	40.0	HS Graduate

In [6]:

```
BA.dropna(subset=["AGE","GENDER","POSITION","EDUCATION","WAGE","HOURS PER WEEK"],inplace=True)
BA.head()
```

Out[6]:

	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATION
0	PID02994	29	male	2015/6/24	Administrative	2015/9/3	15.60	30.0	HS Graduate
1	PID00542	50	male	2015/7/4	Administrative	2015/8/3	8.75	30.0	HS Graduate
2	PID03451	50	male	2014/8/29	Administrative	2015/9/8	16.00	35.0	HS Graduate
3	PID00543	50	male	2015/4/25	Administrative	2015/8/3	8.75	30.0	HS Graduate
4	PID11335	40	male	2015/3/19	Administrative	2015/8/17	13.92	40.0	HS Graduate

In [7]:

```
# Convert the data type of Date
BA["JOB_START_DATE"] = pd.to_datetime(BA["JOB START DATE"])
BA["PROGRAM START DATE"] = pd.to_datetime(BA["PROGRAM START DATE"])
```

In [8]:

```
BA["DayDiff"] = BA["JOB_START_DATE"] - BA["PROGRAM START DATE"]
BA.head()
```

Out[8]:

	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATION
0	PID02994	29	male	2015-06-24	Administrative	2015/9/3	15.60	30.0	HS Graduate
1	PID00542	50	male	2015-07-04	Administrative	2015/8/3	8.75	30.0	HS Graduate
2	PID03451	50	male	2014-08-29	Administrative	2015/9/8	16.00	35.0	HS Graduate
3	PID00543	50	male	2015-04-25	Administrative	2015/8/3	8.75	30.0	HS Graduate
4	PID11335	40	male	2015-03-19	Administrative	2015/8/17	13.92	40.0	HS Graduate

In [9]:

```
BA["DayDiff"] = BA['DayDiff'].dt.days
BA.head()
# DayDiff is how many days for candidates to get a job after submitting application.
```

Out[9]:

	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATION
0	PID02994	29	male	2015-06-24	Administrative	2015/9/3	15.60	30.0	HS Graduate
1	PID00542	50	male	2015-07-04	Administrative	2015/8/3	8.75	30.0	HS Graduate
2	PID03451	50	male	2014-08-29	Administrative	2015/9/8	16.00	35.0	HS Graduate
3	PID00543	50	male	2015-04-25	Administrative	2015/8/3	8.75	30.0	HS Graduate
4	PID11335	40	male	2015-03-19	Administrative	2015/8/17	13.92	40.0	HS Graduate

In [10]:

```
BA.dtypes
```

Out[10]:

```
PID                         object
AGE                          int64
GENDER                      object
PROGRAM START DATE  datetime64[ns]
POSITION                    object
JOB START DATE              object
WAGE                       float64
HOURS PER WEEK             float64
EDUCATION                   object
Quarter and Year            object
JOB_START_DATE      datetime64[ns]
DayDiff                      int64
dtype: object
```

Data Filtering

In [11]:

```
# According to hours per week, we divided dataframe into full-time dataframe and part-time datafram
# Total wages is calculated and added.
FT = BA.loc[BA["HOURS PER WEEK"]>=40.0,:]
FT["TotalWages"] = FT["HOURS PER WEEK"] * FT["WAGE"]
FT.head()
```

```
D:\Python NYU\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
```

Out[11]:

	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATI
4	PID11335	40	male	2015-03-19	Administrative	2015/8/17	13.92	40.0	HS Gradu
9	PID01003	31	female	2014-01-08	Administrative	2014/5/8	12.00	40.0	HS Gradu
11	PID01004	31	female	2014-03-23	Administrative	2014/5/8	12.00	40.0	HS Gradu
19	PID19785	26	female	2015-06-15	Administrative	2015/7/17	10.00	40.0	HS Gradu
21	PID03444	37	female	2014-05-17	Administrative	2014/10/20	13.00	40.0	HS Gradu

In [12]:

```
# According to hours per week, we divided dataframe into full-time dataframe and part-time datafram
# Total wages is calculated and added.
PT = BA.loc[BA["HOURS PER WEEK"]<40.0,:]
PT["TotalWages"] = PT["HOURS PER WEEK"] * PT["WAGE"]
PT.head()
```

```
D:\Python NYU\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
```

Out[12]:

	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATION
0	PID02994	29	male	2015-06-24	Administrative	2015/9/3	15.60	30.0	HS Graduate
1	PID00542	50	male	2015-07-04	Administrative	2015/8/3	8.75	30.0	HS Graduate
2	PID03451	50	male	2014-08-29	Administrative	2015/9/8	16.00	35.0	HS Graduate
3	PID00543	50	male	2015-04-25	Administrative	2015/8/3	8.75	30.0	HS Graduate
5	PID05307	26	male	2015-04-05	Administrative	2015/9/16	16.50	35.0	HS Graduate

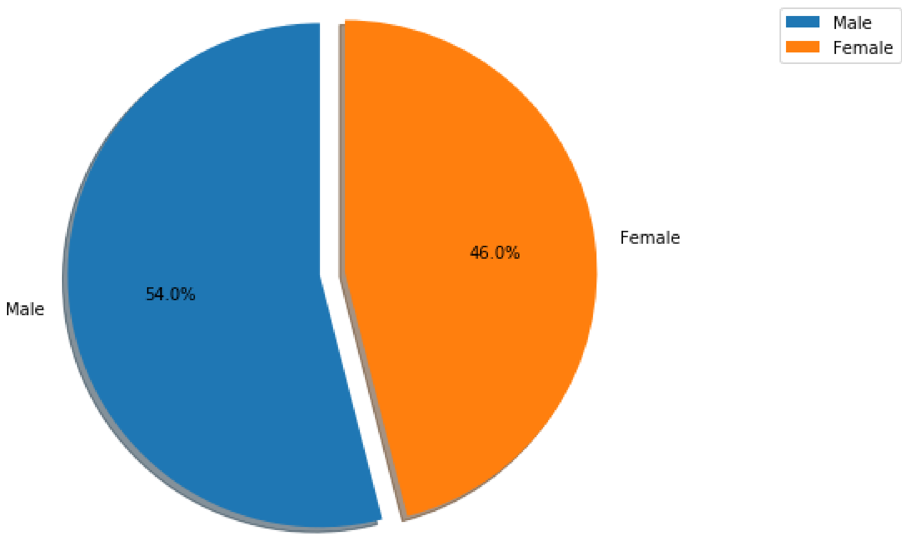
1.Descriptive Data Mining

1.1 Data Visualization

Gender

In [13]:

```
explode = (0.1,0)
fig1, ax1 = plt.subplots(figsize=(10,5))
ax1.pie(BA['GENDER'].value_counts(), explode=explode,labels=['Male','Female'], autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.tight_layout()
plt.legend()
plt.show()
```



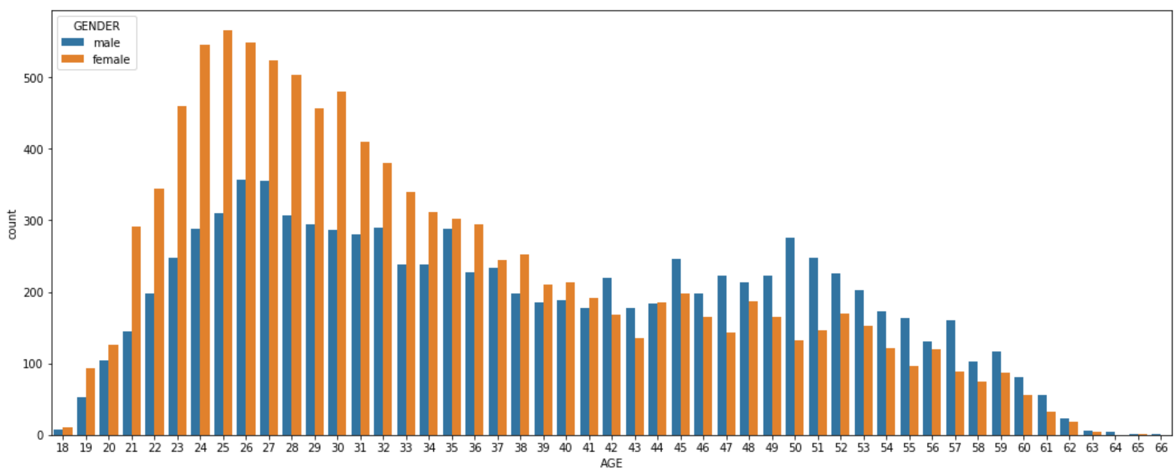
Age

In [14]:

```
fig1, ax1 = plt.subplots(figsize=(18,7))
sns.countplot(BA['AGE'],hue=BA['GENDER'])
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f1cb10f5f8>
```



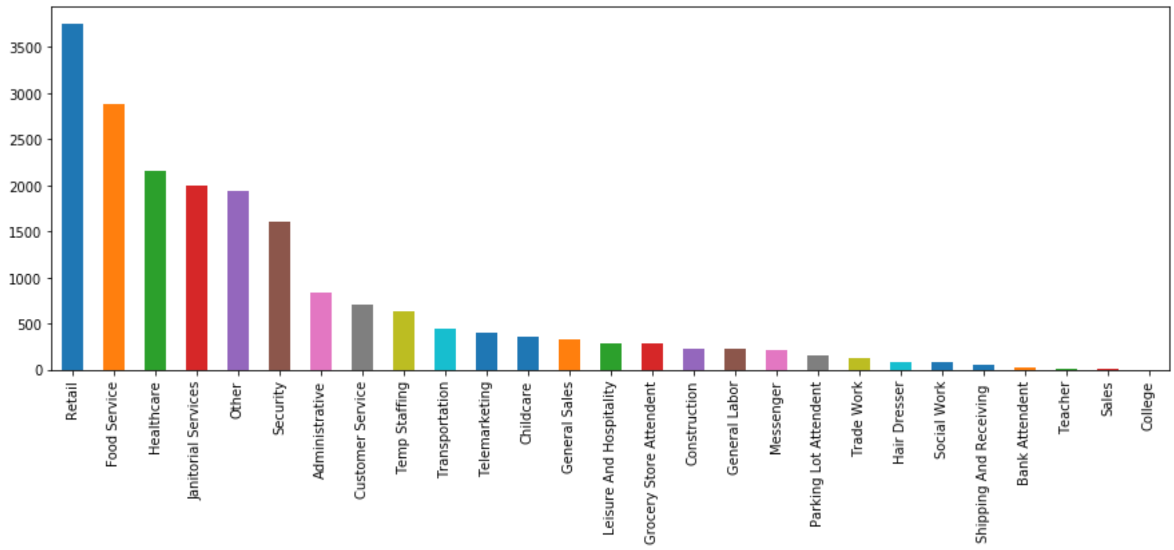
Position

In [15]:

```
BA['POSITION'].value_counts().plot.bar(figsize=(15,5))
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f1cb2d0588>
```



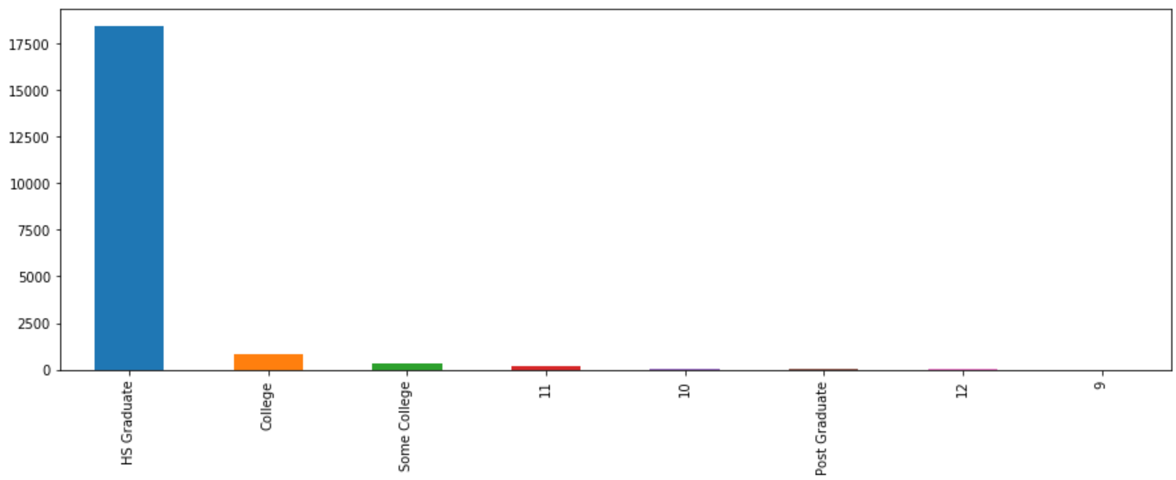
Education

In [16]:

```
BA['EDUCATION'].value_counts().plot.bar(figsize=(15,5))
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f1cb399e10>
```



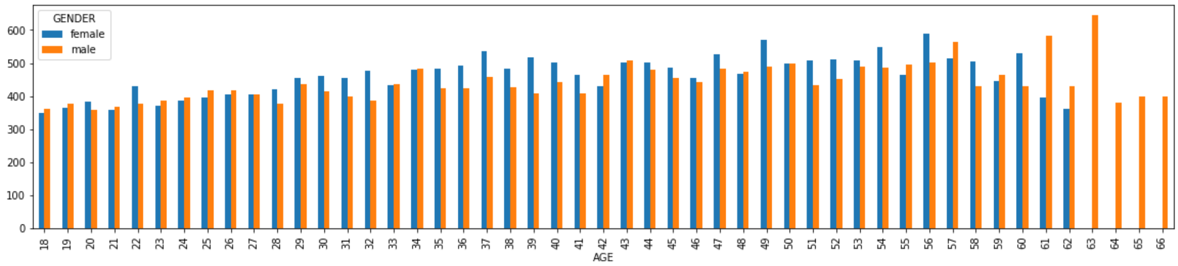
We can use below code to combine or groupby any variables to observe

In [17]:

```
FT.pivot_table(values='TotalWages',index='AGE',columns='GENDER').plot.bar(figsize=(20,4))
#Full-time Total Wages per week
```

Out[17]:

<matplotlib.axes._subplots.AxesSubplot at 0x1f1cb411390>

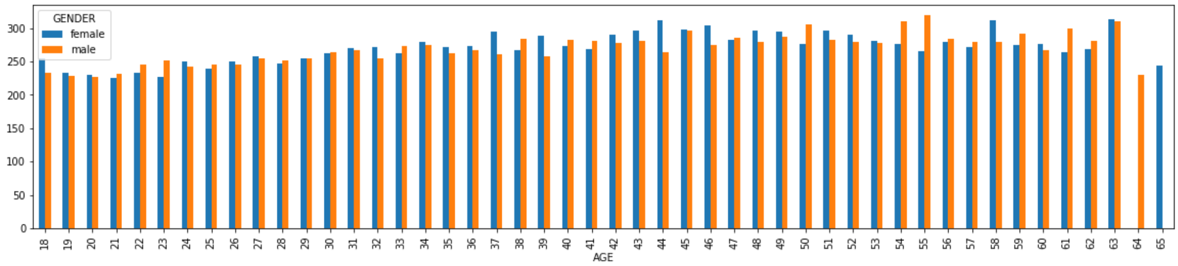


In [18]:

```
PT.pivot_table(values='TotalWages',index='AGE',columns='GENDER').plot.bar(figsize=(20,4))
# Part-time Total Wages per week
```

Out[18]:

<matplotlib.axes._subplots.AxesSubplot at 0x1f1cb3f1128>

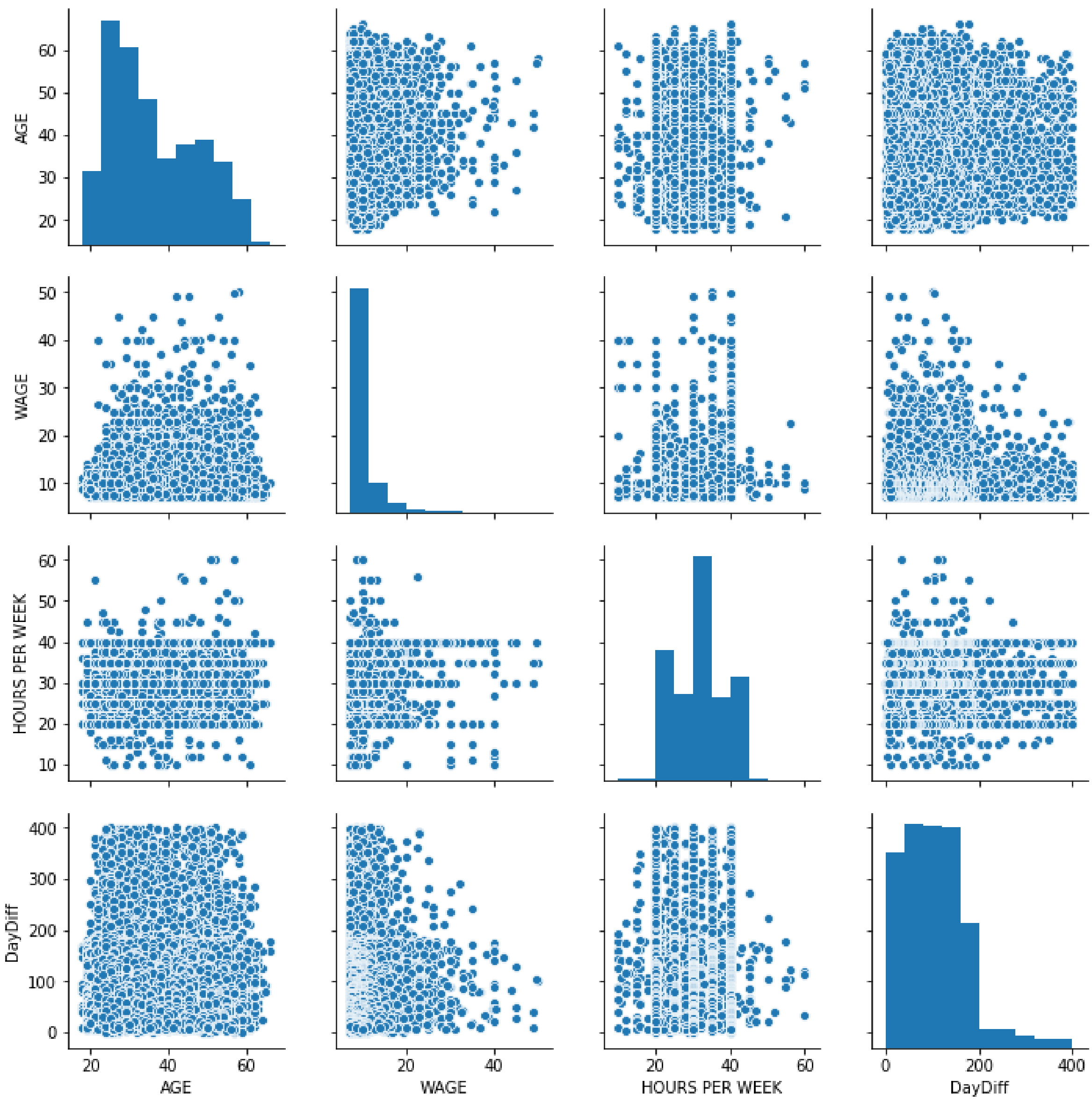


In [19]:

```
sns.pairplot(BA)
# We can see the correltation between each variable.
# It does not make sense as most of them are categorical data
```

Out[19]:

<seaborn.axisgrid.PairGrid at 0x1f1c99fd710>



1.2 K-Means

In [20]:

```
from sklearn.cluster import KMeans
```

In [21]:

```
X = PT[["WAGE","HOURS PER WEEK"]]
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
kmeans
```

Out[21]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

In [22]:

```
kmeans.labels_
# We can determine it belongs to which group
```

Out[22]:

```
array([2, 0, 2, ..., 1, 2, 0])
```

In [23]:

```
y_kmeans = kmeans.predict([[15.8, 20], [20, 40]])
y_kmeans
# We can predict it belongs to which group
```

Out[23]:

```
array([1, 2])
```

In [24]:

```
kmeans.cluster_centers_
```

Out[24]:

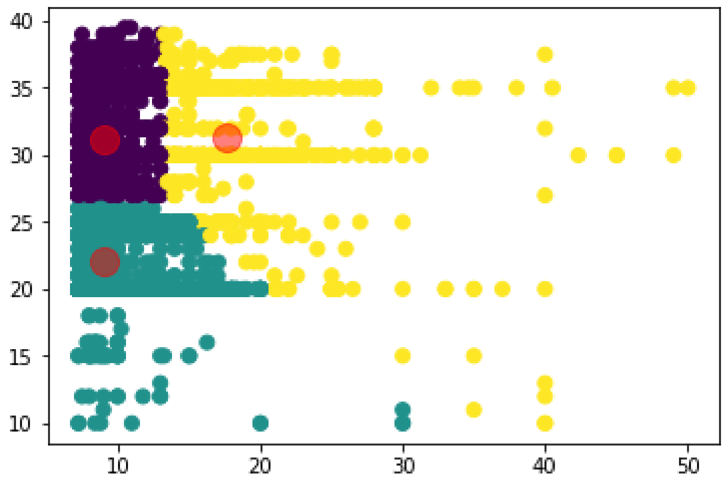
```
array([[ 8.99078275, 31.25124175],
       [ 9.0519841 , 22.09444444],
       [17.55978031, 31.40489914]])
```

In [25]:

```
x = np.array(X)
y_kmeans = kmeans.predict(x)

plt.scatter(x[:, 0], x[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.5)
```

Out[25]:

<matplotlib.collections.PathCollection at 0x1f1ce95fc18>



In overall, K-means does not help us to do predictive data mining. Because we already have columns to seperate these data as different independent variables, we just show what we did.

2. Predictive Data Mining

2.1 Prediction for Numerical Data

2.1.1 Full-Time DayDiff & Part-Time DayDiff

2.1.1 a) Linear Regression

In [26]:

```
FT.head()
```

Out[26]:

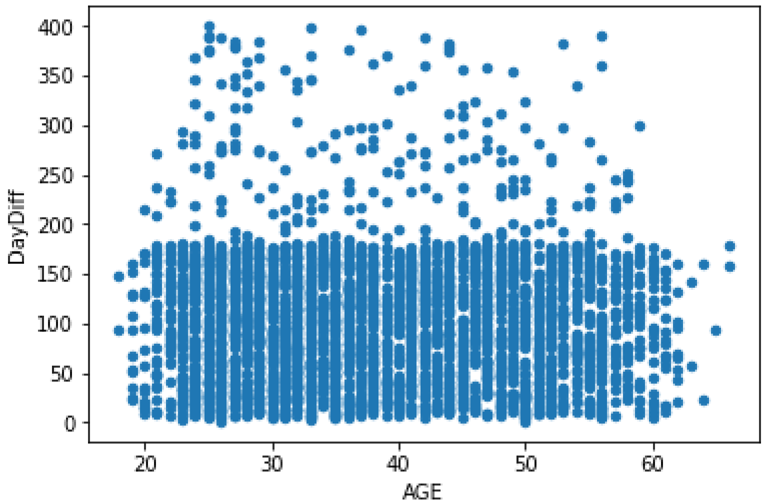
	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATI(
4	PID11335	40	male	2015-03-19	Administrative	2015/8/17	13.92	40.0	HS Gradu:
9	PID01003	31	female	2014-01-08	Administrative	2014/5/8	12.00	40.0	HS Gradu:
11	PID01004	31	female	2014-03-23	Administrative	2014/5/8	12.00	40.0	HS Gradu:
19	PID19785	26	female	2015-06-15	Administrative	2015/7/17	10.00	40.0	HS Gradu:
21	PID03444	37	female	2014-05-17	Administrative	2014/10/20	13.00	40.0	HS Gradu:

In [27]:

```
FT.plot.scatter(y='DayDiff',x='AGE')
# The relationship between age and daydiff is random
```

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x1f1ce985668>



There is no need to show scatterplot for categorical data

In [28]:

```
import statsmodels.formula.api as smf
```

In [29]:

```
reg_DayDiff = smf.ols('DayDiff~ AGE + GENDER + EDUCATION + POSITION ',data=FT).fit()
print(reg_DayDiff.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                DayDiff   R-squared:                       0.011
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     1.024
Date:                Thu, 16 May 2019   Prob (F-statistic):              0.430
Time:                        01:45:05   Log-Likelihood:                -18429.
No. Observations:                3276   AIC:                         3.693e+04
Df Residuals:                    3241   BIC:                         3.714e+04
Df Model:                          34
Covariance Type:            nonrobust
==========================================================================================
===================
                                          coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------------------
-------------------
Intercept                              85.2320     22.507      3.787      0.000
41.103     129.361
GENDER[T.male]                          6.7339      2.571      2.619      0.009
1.693      11.774
EDUCATION[T.11]                       -18.8170     27.661     -0.680      0.496
-73.053      35.419
EDUCATION[T.12]                        10.7643     52.403      0.205      0.837
-91.981     113.510
EDUCATION[T.College]                    9.0941     21.994      0.413      0.679
-34.030      52.218
EDUCATION[T.HS Graduate]               10.4506     21.464      0.487      0.626
-31.634      52.535
EDUCATION[T.Post Graduate]             38.5699     30.446      1.267      0.205
-21.125      98.265
EDUCATION[T.Some College]               0.0480     22.733      0.002      0.998
-44.525      44.621
POSITION[T.Bank Attendent]              3.6433     19.293      0.189      0.850
-34.184      41.470
POSITION[T.Childcare]                  10.6650     10.893      0.979      0.328
-10.692      32.022
POSITION[T.College]                    79.3201     67.768      1.170      0.242
-53.553     212.194
POSITION[T.Construction]               16.7188      9.121      1.833      0.067
-1.165      34.603
POSITION[T.Customer Service]            8.9634      8.419      1.065      0.287
-7.544      25.471
POSITION[T.Food Service]                4.7790      5.958      0.802      0.423
-6.902      16.460
POSITION[T.General Labor]              13.2798     12.784      1.039      0.299
-11.786      38.345
POSITION[T.General Sales]               6.5114      8.839      0.737      0.461
-10.818      23.841
POSITION[T.Grocery Store Attendent]    11.5520     12.306      0.939      0.348
-12.576      35.680
POSITION[T.Hair Dresser]               50.8431     39.252      1.295      0.195
-26.118     127.804
POSITION[T.Healthcare]                 12.2880      6.019      2.041      0.041
```

```
0.486      24.090
POSITION[T.Janitorial Services]        10.9586      5.713      1.918      0.055
-0.242      22.159
POSITION[T.Leisure And Hospitality]    11.1044      9.417      1.179      0.238
-7.359      29.568
POSITION[T.Messenger]                   9.4401     10.806      0.874      0.382
-11.747      30.627
POSITION[T.Other]                       2.6076      5.578      0.467      0.640
-8.330      13.545
POSITION[T.Parking Lot Attendent]       2.4290     10.993      0.221      0.825
-19.124      23.982
POSITION[T.Retail]                      9.5035      5.553      1.711      0.087
-1.384      20.391
POSITION[T.Sales]                      20.9074     67.655      0.309      0.757    -
111.743      153.558
POSITION[T.Security]                   13.9615      6.177      2.260      0.024
1.851      26.072
POSITION[T.Shipping And Receiving ]    14.7305     20.045      0.735      0.462
-24.572      54.033
POSITION[T.Social Work]                -3.3898     13.977     -0.243      0.808
-30.795      24.015
POSITION[T.Teacher]                    -9.4907     39.228     -0.242      0.809
-86.404      67.423
POSITION[T.Telemarketing]              12.2516     13.761      0.890      0.373
-14.730      39.233
POSITION[T.Temp Staffing]              -0.2985      8.309     -0.036      0.971
-16.591      15.994
POSITION[T.Trade Work]                 18.6984     10.043      1.862      0.063
-0.993      38.390
POSITION[T.Transportation]              2.7367      7.361      0.372      0.710
-11.696      17.169
AGE                                    -0.0704      0.113     -0.621      0.535
-0.293      0.152
==============================================================================
Omnibus:                      605.238   Durbin-Watson:                   0.482
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1280.877
Skew:                           1.077   Prob(JB):                    7.26e-279
Kurtosis:                       5.178   Cond. No.                     2.31e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 2.31e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

There is no linear regression as most of variable of p-value is very high.

In [30]:

```
PT.head()
```

Out[30]:

	PID	AGE	GENDER	PROGRAM START DATE	POSITION	JOB START DATE	WAGE	HOURS PER WEEK	EDUCATION
0	PID02994	29	male	2015-06-24	Administrative	2015/9/3	15.60	30.0	HS Graduate
1	PID00542	50	male	2015-07-04	Administrative	2015/8/3	8.75	30.0	HS Graduate
2	PID03451	50	male	2014-08-29	Administrative	2015/9/8	16.00	35.0	HS Graduate
3	PID00543	50	male	2015-04-25	Administrative	2015/8/3	8.75	30.0	HS Graduate
5	PID05307	26	male	2015-04-05	Administrative	2015/9/16	16.50	35.0	HS Graduate

In [31]:

```
reg_DayDiff = smf.ols('DayDiff~ AGE + GENDER + EDUCATION + POSITION ',data=PT).fit()
print(reg_DayDiff.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                DayDiff   R-squared:                       0.003
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.476
Date:                Thu, 16 May 2019   Prob (F-statistic):             0.0365
Time:                        01:45:06   Log-Likelihood:                -94031.
No. Observations:               16613   AIC:                         1.881e+05
Df Residuals:                   16578   BIC:                         1.884e+05
Df Model:                          34
Covariance Type:            nonrobust
==========================================================================================
===================
                                          coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------------------
-------------------
Intercept                              98.2567      8.813     11.150      0.000
80.983     115.530
GENDER[T.male]                         -0.1868      1.169     -0.160      0.873
-2.478       2.105
EDUCATION[T.11]                         0.8677      9.818      0.088      0.930
-18.377      20.112
EDUCATION[T.12]                        15.0531     17.235      0.873      0.382
-18.730      48.836
EDUCATION[T.9]                         -3.8855     25.928     -0.150      0.881
-54.707      46.936
EDUCATION[T.College]                   -2.4576      8.612     -0.285      0.775
-19.338      14.423
EDUCATION[T.HS Graduate]                2.6952      8.173      0.330      0.742
-13.325      18.715
EDUCATION[T.Post Graduate]             11.8821     18.775      0.633      0.527
-24.918      48.682
EDUCATION[T.Some College]              -4.8975      9.348     -0.524      0.600
-23.220      13.425
POSITION[T.Bank Attendent]              6.6036     17.622      0.375      0.708
-27.938      41.145
POSITION[T.Childcare]                   4.7234      4.873      0.969      0.332
-4.828      14.275
POSITION[T.Construction]               17.7776      6.185      2.874      0.004
5.655      29.900
POSITION[T.Customer Service]            7.6673      3.973      1.930      0.054
-0.121      15.455
POSITION[T.Food Service]                1.6167      3.170      0.510      0.610
-4.597       7.831
POSITION[T.General Labor]               4.3964      5.737      0.766      0.443
-6.848      15.641
POSITION[T.General Sales]               4.1324      5.250      0.787      0.431
-6.159      14.423
POSITION[T.Grocery Store Attendent]     7.6705      5.241      1.463      0.143
-2.603      17.944
POSITION[T.Hair Dresser]                4.6864      8.026      0.584      0.559
-11.046      20.419
POSITION[T.Healthcare]                  4.8166      3.271      1.473      0.141
```

```
-1.595      11.228
POSITION[T.Janitorial Services]          3.0470      3.327      0.916      0.360
-3.474      9.568
POSITION[T.Leisure And Hospitality]     11.6624      5.463      2.135      0.033
0.954       22.371
POSITION[T.Messenger]                   20.8549      6.185      3.372      0.001
8.731       32.979
POSITION[T.Other]                        3.5736      3.353      1.066      0.287
-2.999      10.146
POSITION[T.Parking Lot Attendent]       -5.4787      7.150     -0.766      0.444
-19.493      8.536
POSITION[T.Retail]                       3.9535      3.102      1.275      0.202
-2.126      10.033
POSITION[T.Sales]                       -4.2654     28.547     -0.149      0.881
-60.220      51.689
POSITION[T.Security]                     9.7807      3.417      2.862      0.004
3.083       16.479
POSITION[T.Shipping And Receiving ]      8.8986     11.233      0.792      0.428
-13.119      30.916
POSITION[T.Social Work]                  3.4198      9.278      0.369      0.712
-14.766      21.605
POSITION[T.Teacher]                     10.7540     18.189      0.591      0.554
-24.898      46.406
POSITION[T.Telemarketing]               -0.4157      4.587     -0.091      0.928
-9.408      8.576
POSITION[T.Temp Staffing]                4.4134      4.110      1.074      0.283
-3.643      12.470
POSITION[T.Trade Work]                  -2.2903      8.734     -0.262      0.793
-19.410      14.830
POSITION[T.Transportation]              12.4824      4.871      2.563      0.010
2.935       22.030
AGE                                      0.0091      0.051      0.177      0.859
-0.092       0.110
==============================================================================
Omnibus:                     3117.909   Durbin-Watson:                   0.412
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6416.319
Skew:                           1.118   Prob(JB):                         0.00
Kurtosis:                       5.066   Cond. No.                     1.98e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 1.98e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

There is no linear relationship as most of variable of p-value is very high.

Since there is no linear relationship between independent varibale and DayDiff, we will not show regression tree model either due to much weaker performance of regression tree if we use, as well as we will not show score of this model

2.1.2 Full-Time TotalWages

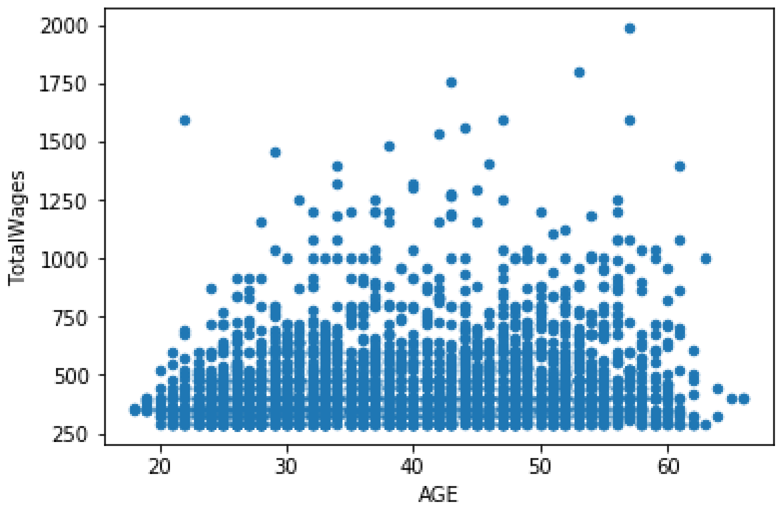
2.1.2 a) Linear Regression

In [32]:

```
FT.plot.scatter(y='TotalWages',x='AGE')
# The relationship between age and Totalwages is random
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f1ced09dd8>
```



In [33]:

```
reg_PTWages = smf.ols('TotalWages ~ AGE + GENDER + EDUCATION + POSITION ',data=FT).fit()
print(reg_PTWages.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             TotalWages   R-squared:                       0.357
Model:                            OLS   Adj. R-squared:                  0.350
Method:                 Least Squares   F-statistic:                     52.86
Date:                Thu, 16 May 2019   Prob (F-statistic):          2.76e-280
Time:                        01:45:06   Log-Likelihood:                -20907.
No. Observations:                3276   AIC:                         4.188e+04
Df Residuals:                    3241   BIC:                         4.210e+04
Df Model:                          34
Covariance Type:            nonrobust
==========================================================================================
===================
                                           coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------------------
-------------------
Intercept                              519.6937     47.964     10.835      0.000
425.650     613.737
GENDER[T.male]                         -10.0986      5.479     -1.843      0.065
-20.840       0.643
EDUCATION[T.11]                        -10.4827     58.949     -0.178      0.859     -
126.064     105.098
EDUCATION[T.12]                          4.9201    111.674      0.044      0.965     -
214.040     223.880
EDUCATION[T.College]                   216.9820     46.872      4.629      0.000
125.080     308.884
EDUCATION[T.HS Graduate]                83.3478     45.742      1.822      0.069
-6.338     173.034
EDUCATION[T.Post Graduate]             644.8053     64.883      9.938      0.000
517.590     772.021
EDUCATION[T.Some College]              196.7548     48.446      4.061      0.000
101.766     291.744
POSITION[T.Bank Attendent]            -115.5768     41.114     -2.811      0.005     -
196.190     -34.964
POSITION[T.Childcare]                 -272.0651     23.213    -11.720      0.000     -
317.579    -226.551
POSITION[T.College]                    171.3651    144.421      1.187      0.235     -
111.800     454.530
POSITION[T.Construction]              -102.1791     19.438     -5.257      0.000     -
140.291     -64.067
POSITION[T.Customer Service]          -249.1236     17.942    -13.885      0.000     -
284.303    -213.945
POSITION[T.Food Service]              -291.5874     12.696    -22.967      0.000     -
316.481    -266.694
POSITION[T.General Labor]             -284.6089     27.244    -10.447      0.000     -
338.026    -231.192
POSITION[T.General Sales]             -247.7015     18.836    -13.150      0.000     -
284.633    -210.770
POSITION[T.Grocery Store Attendent]   -320.3279     26.225    -12.215      0.000     -
371.747    -268.909
POSITION[T.Hair Dresser]              -297.4295     83.649     -3.556      0.000     -
461.440    -133.419
POSITION[T.Healthcare]                -207.5782     12.828    -16.182      0.000     -
```

```
232.729    -182.427
POSITION[T.Janitorial Services]      -290.4554     12.174    -23.859      0.000    -
314.325    -266.586
POSITION[T.Leisure And Hospitality]  -239.4661     20.068    -11.933      0.000    -
278.813    -200.119
POSITION[T.Messenger]                -297.2100     23.028    -12.906      0.000    -
342.362    -252.058
POSITION[T.Other]                    -240.7650     11.888    -20.253      0.000    -
264.074    -217.456
POSITION[T.Parking Lot Attendent]    -317.4066     23.426    -13.549      0.000    -
363.339    -271.475
POSITION[T.Retail]                   -294.2677     11.834    -24.866      0.000    -
317.471    -271.065
POSITION[T.Sales]                    -343.1077    144.178     -2.380      0.017    -
625.797     -60.418
POSITION[T.Security]                 -276.0624     13.163    -20.972      0.000    -
301.872    -250.253
POSITION[T.Shipping And Receiving ]  -210.5613     42.718     -4.929      0.000    -
294.318    -126.804
POSITION[T.Social Work]               -81.8688     29.787     -2.748      0.006    -
140.272     -23.466
POSITION[T.Teacher]                  -176.6394     83.598     -2.113      0.035    -
340.549     -12.730
POSITION[T.Telemarketing]            -331.2746     29.326    -11.296      0.000    -
388.775    -273.774
POSITION[T.Temp Staffing]            -252.0669     17.708    -14.235      0.000    -
286.787    -217.347
POSITION[T.Trade Work]               -105.9514     21.403     -4.950      0.000    -
147.916     -63.987
POSITION[T.Transportation]           -237.7526     15.687    -15.156      0.000    -
268.510    -206.995
AGE                                     2.1262      0.242      8.800      0.000
1.652       2.600
==============================================================================
Omnibus:                     1357.433   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11228.498
Skew:                           1.755   Prob(JB):                         0.00
Kurtosis:                      11.363   Cond. No.                     2.31e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 2.31e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The P-value of Gender is very high, so we just remove this variable

In [34]:

```
reg_FTWages = smf.ols('TotalWages ~ AGE + EDUCATION + POSITION ',data=FT).fit()
print(reg_FTWages.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:             TotalWages   R-squared:                       0.356
Model:                            OLS   Adj. R-squared:                  0.349
Method:                 Least Squares   F-statistic:                     54.32
Date:                Thu, 16 May 2019   Prob (F-statistic):          1.97e-280
Time:                        01:45:06   Log-Likelihood:                -20909.
No. Observations:                3276   AIC:                         4.189e+04
Df Residuals:                    3242   BIC:                         4.209e+04
Df Model:                          33                                         
Covariance Type:            nonrobust                                         
====================================================================================
===================
                                        coef    std err          t      P>|t|      
[0.025      0.975]
------------------------------------------------------------------------------------
-------------------
Intercept                           519.8052     47.982     10.833      0.000     
425.727     613.883
EDUCATION[T.11]                     -12.9170     58.956     -0.219      0.827     -
128.512     102.678
EDUCATION[T.12]                       4.2152    111.715      0.038      0.970     -
214.824     223.255
EDUCATION[T.College]                214.9666     46.877      4.586      0.000     
123.056     306.877
EDUCATION[T.HS Graduate]             81.8429     45.752      1.789      0.074     
-7.862     171.548
EDUCATION[T.Post Graduate]          643.1119     64.900      9.909      0.000     
515.862     770.362
EDUCATION[T.Some College]           195.2462     48.457      4.029      0.000     
100.236     290.256
POSITION[T.Bank Attendent]         -111.6701     41.075     -2.719      0.007     -
192.206     -31.135
POSITION[T.Childcare]              -268.4820     23.140    -11.602      0.000     -
313.853    -223.111
POSITION[T.College]                 167.0810    144.455      1.157      0.248     -
116.152     450.314
POSITION[T.Construction]           -105.4834     19.363     -5.448      0.000     -
143.447     -67.519
POSITION[T.Customer Service]       -249.0833     17.949    -13.878      0.000     -
284.275    -213.891
POSITION[T.Food Service]           -293.8012     12.644    -23.237      0.000     -
318.592    -269.010
POSITION[T.General Labor]          -285.5866     27.249    -10.481      0.000     -
339.013    -232.160
POSITION[T.General Sales]          -248.7587     18.834    -13.208      0.000     -
285.687    -211.831
POSITION[T.Grocery Store Attendent] -321.9777     26.219    -12.280      0.000     -
373.385    -270.570
POSITION[T.Hair Dresser]           -292.3427     83.634     -3.495      0.000     -
456.324    -128.361
POSITION[T.Healthcare]             -205.0251     12.757    -16.071      0.000     -
230.038    -180.012
POSITION[T.Janitorial Services]    -292.7800     12.113    -24.171      0.000     -
```

```
316.530     -269.030
POSITION[T.Leisure And Hospitality]  -240.9428      20.059     -12.011        0.000     -
280.273     -201.612
POSITION[T.Messenger]                -300.7718      22.956     -13.102        0.000     -
345.781     -255.763
POSITION[T.Other]                    -242.3823      11.860     -20.437        0.000     -
265.636     -219.128
POSITION[T.Parking Lot Attendent]    -321.2310      23.343     -13.761        0.000     -
366.999     -275.463
POSITION[T.Retail]                   -295.5186      11.819     -25.004        0.000     -
318.692     -272.345
POSITION[T.Sales]                    -349.1243     144.194      -2.421        0.016     -
631.846      -66.403
POSITION[T.Security]                 -278.3872      13.108     -21.238        0.000     -
304.088     -252.687
POSITION[T.Shipping And Receiving ]  -209.8728      42.732      -4.911        0.000     -
293.658     -126.088
POSITION[T.Social Work]               -81.8933      29.798      -2.748        0.006     -
140.318      -23.469
POSITION[T.Teacher]                  -175.4077      83.626      -2.098        0.036     -
339.373      -11.443
POSITION[T.Telemarketing]            -331.2366      29.337     -11.291        0.000     -
388.758     -273.715
POSITION[T.Temp Staffing]            -253.0177      17.707     -14.289        0.000     -
287.736     -218.299
POSITION[T.Trade Work]               -108.5707      21.364      -5.082        0.000     -
150.458      -66.683
POSITION[T.Transportation]           -239.5564      15.662     -15.295        0.000     -
270.265     -208.848
AGE                                     2.0447       0.238       8.605        0.000
1.579        2.511
==============================================================================
Omnibus:                     1359.532   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11188.373
Skew:                           1.760   Prob(JB):                         0.00
Kurtosis:                      11.341   Cond. No.                     2.31e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 2.31e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Education[11], Education[12], Education[HS Graduate], and Position[College] is not significant since p-value is very high.

Therefore, we can get that PTWages = 2.0447* AGE - b1 * Position** - b2 * Education** + 519.8052

Positon** and Education** are categorical data, so we need to match it with different coefficient. Also, for Education with 11,12, HS Graduate and Position with College, this regression has no effect.

In [35]:

```
from sklearn.linear_model import LinearRegression as reg
from patsy import dmatrices
```

In [36]:

```
y, X = dmatrices("TotalWages ~ AGE + POSITION + EDUCATION", data = FT)
```

In [37]:

```
reg().fit(X, y).score(X, y)
```

Out[37]:

0.3560325587063272

2.1.2 b) Regression Tree

In [38]:

```
Features1 = FT[["GENDER","POSITION","EDUCATION"]].values
```

In [39]:

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()

encFeatures1 = enc.fit_transform(Features1).toarray()
encFeatures1 = pd.DataFrame(data=encFeatures1)
encFeatures1["AGE"] = list(FT["AGE"])
encFeatures1.head()
```

Out[39]:

	0	1	2	3	4	5	6	7	8	9	...	27	28	29	30	31	32	33	34	35
0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
2	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0

5 rows × 37 columns

In [40]:

```
TotalWages1 = FT[["TotalWages"]]
```

In [41]:

```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(encFeatures1, TotalWages1, random_state=33)
```

In [42]:

```
from sklearn.tree import DecisionTreeRegressor
Reg_tree = DecisionTreeRegressor()
```

In [43]:

```
Reg_tree.fit(X_train1,y_train1)
```

Out[43]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [44]:

```
X_test1.head()
```

Out[44]:

	0	1	2	3	4	5	6	7	8	9	...	27	28	29	30	31	32	33	34	3
125	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0
228	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0
2335	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0
726	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0
1113	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0

5 rows × 37 columns

In [45]:

```
Reg_tree.predict(X_test1)
```

Out[45]:

```
array([ 880.        , 1125.        ,  372.22222222,  358.33333333,
        520.4       ,  495.06666667,  418.57142857,  720.        ,
        330.        ,  400.        ,  601.2       ,  382.        ,
        400.        ,  640.        ,  333.5       ,  574.8       ,
        345.        ,  410.        ,  314.44444444,  508.212     ,
        376.        ,  423.33333333,  540.        ,  448.57142857,
        448.        ,  842.        ,  400.        ,  500.        ,
        358.        ,  421.42857143,  350.        ,  360.        ,
        380.        ,  542.11428571,  550.        ,  372.22222222,
        960.        ,  448.88888889,  374.        ,  406.66666667,
        360.        ,  655.6       ,  429.11111111,  455.        ,
        326.66666667,  400.        ,  336.66666667,  440.        ,
        378.        ,  429.11111111,  420.        ,  740.        ,
        354.33333333,  660.        ,  444.66666667,  360.        ,
        480.        ,  420.        ,  426.        ,  373.33333333,
        350.        ,  378.        ,  470.        ,  392.33333333,
        475.        ,  500.        ,  350.        ,  315.        ,
```

In [46]:

```
Reg_tree.score(X_test1,y_test1)
```

Out[46]:

```
0.043898369897300005
```

In [47]:

```
scorelist_regFTW =[]
length_regFTW = len(scorelist_regFTW)

#use for loop to list 20 scores calculated randomly by regression tree
for length_regFTW in range(1,21):
    from sklearn.tree import DecisionTreeRegressor
    Reg_tree_FTW = DecisionTreeRegressor()
    Reg_tree_FTW.fit(X_train1,y_train1)

    scorelist_regFTW.append(Reg_tree_FTW.score(X_test1,y_test1))
```

In [48]:

```
scorelist_regFTW
```

Out[48]:

```
[0.02752729284538813,
 0.027504220984649552,
 0.04343616980338283,
 0.030581158390299512,
 0.0371123509301 8316,
 0.03241609095969089,
 0.04135247378773898,
 0.036855294539211436,
 0.03173509927599494,
 0.05191293517345319,
 0.027425630345445984,
 0.050014027334125515,
 0.04014056870701155,
 0.03201723359999442,
 0.0314390576128335,
 0.03883903312691539,
 0.02679247456034861,
 0.020840042422896277,
 0.02843699775073938,
 0.029386589307524913]
```

In [49]:

```
#Compute average score
np.mean(scorelist_regFTW)
```

Out[49]:

```
0.03428823707289141
```

For Full-Time TotalWages, we should use linear regression model as the score of linear regression (0.35603) is higher than the average score of regression tree (0.03236)

When we have information of gender, age, position, and education, we can use multivariable linear regression model to predict wages for full-time employees.

2.1.3 Part-Time TotalWages

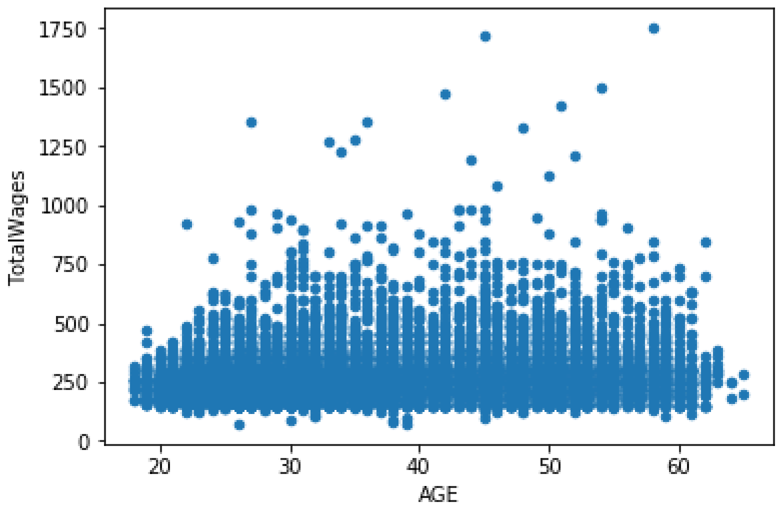
2.1.3 a) Linear Regression

In [50]:

```
PT.plot.scatter(y='TotalWages',x='AGE')
```

Out[50]:

<matplotlib.axes._subplots.AxesSubplot at 0x1f1cbca9668>



In [51]:

```
reg_FTWages = smf.ols('TotalWages ~ AGE + GENDER + EDUCATION + POSITION ',data=PT).fit()
print(reg_FTWages.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             TotalWages   R-squared:                       0.221
Model:                            OLS   Adj. R-squared:                  0.219
Method:                 Least Squares   F-statistic:                     138.1
Date:                Thu, 16 May 2019   Prob (F-statistic):               0.00
Time:                        01:45:08   Log-Likelihood:                -98238.
No. Observations:               16613   AIC:                         1.965e+05
Df Residuals:                   16578   BIC:                         1.968e+05
Df Model:                          34
Covariance Type:            nonrobust
====================================================================================
===================
                                       coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------------
-------------------
Intercept                          312.8936     11.352     27.562      0.000
290.642     335.145
GENDER[T.male]                       3.0075      1.506      1.997      0.046
0.056       5.960
EDUCATION[T.11]                     13.5516     12.648      1.071      0.284
-11.240      38.343
EDUCATION[T.12]                      9.1553     22.202      0.412      0.680
-34.364      52.674
EDUCATION[T.9]                      -9.0542     33.401     -0.271      0.786
-74.523      56.415
EDUCATION[T.College]               108.1635     11.094      9.750      0.000
86.418     129.909
EDUCATION[T.HS Graduate]            40.0675     10.528      3.806      0.000
19.431      60.704
EDUCATION[T.Post Graduate]         534.7921     24.185     22.112      0.000
487.386     582.198
EDUCATION[T.Some College]          120.2649     12.041      9.988      0.000
96.662     143.868
POSITION[T.Bank Attendent]        -103.0031     22.701     -4.537      0.000     -
147.500     -58.506
POSITION[T.Childcare]             -134.3139      6.277    -21.397      0.000     -
146.618    -122.010
POSITION[T.Construction]           -33.9372      7.967     -4.260      0.000
-49.554     -18.321
POSITION[T.Customer Service]       -96.8482      5.118    -18.922      0.000     -
106.881     -86.816
POSITION[T.Food Service]          -143.9380      4.084    -35.244      0.000     -
151.943    -135.933
POSITION[T.General Labor]         -133.1446      7.390    -18.016      0.000     -
147.630    -118.659
POSITION[T.General Sales]          -68.2561      6.763    -10.092      0.000
-81.513     -54.999
POSITION[T.Grocery Store Attendent] -144.6497    6.752    -21.424      0.000     -
157.884    -131.415
POSITION[T.Hair Dresser]          -169.4499     10.340    -16.389      0.000     -
189.716    -149.183
POSITION[T.Healthcare]            -108.3047      4.214    -25.704      0.000     -
```

```
116.564    -100.046
POSITION[T.Janitorial Services]      -135.3663      4.286     -31.587      0.000     -
143.766    -126.966
POSITION[T.Leisure And Hospitality]  -106.0744      7.038     -15.072      0.000     -
119.869     -92.279
POSITION[T.Messenger]                -148.3747      7.968     -18.621      0.000     -
163.993    -132.757
POSITION[T.Other]                    -113.0094      4.320     -26.162      0.000     -
121.476    -104.543
POSITION[T.Parking Lot Attendent]    -125.6311      9.211     -13.640      0.000     -
143.685    -107.578
POSITION[T.Retail]                   -157.9162      3.996     -39.521      0.000     -
165.748    -150.084
POSITION[T.Sales]                    -150.6998     36.774      -4.098      0.000     -
222.781     -78.619
POSITION[T.Security]                 -117.6954      4.402     -26.736      0.000     -
126.324    -109.067
POSITION[T.Shipping And Receiving ]   -79.3162     14.470      -5.481      0.000     -
107.679     -50.954
POSITION[T.Social Work]                28.8185     11.952       2.411      0.016
5.392      52.245
POSITION[T.Teacher]                   -17.4748     23.431      -0.746      0.456
-63.402      28.452
POSITION[T.Telemarketing]            -160.3357      5.910     -27.132      0.000     -
171.919    -148.752
POSITION[T.Temp Staffing]            -121.6684      5.295     -22.979      0.000     -
132.046    -111.290
POSITION[T.Trade Work]                 25.2399     11.252       2.243      0.025
3.186      47.294
POSITION[T.Transportation]            -99.7980      6.275     -15.905      0.000     -
112.097     -87.499
AGE                                     0.8990      0.066      13.555      0.000
0.769       1.029
==============================================================================
Omnibus:                     8259.763   Durbin-Watson:                   2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           124512.759
Skew:                           2.021   Prob(JB):                         0.00
Kurtosis:                      15.788   Cond. No.                     1.98e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 1.98e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Education[9],Education[11], Education[12],Position[Teacher] is not significant since p-value is very high.

Therefore, we can get that PTWages = 0.8990 * AGE + 3.0075 * Gender - b1 * Position** - b2 * Education** + 312.8936

Positon** and Education** are categorical data, so we need to match it with different coefficient. Also, for Education with 9,11,12 and Position with Teacher, this regression has no effect.

In [52]:

```
from sklearn.linear_model import LinearRegression as reg
from patsy import dmatrices
```

In [53]:

```
y, X = dmatrices("TotalWages ~ GENDER + AGE + POSITION + EDUCATION",data = PT)
```

In [54]:

```
reg().fit(X,y).score(X,y)
```

Out[54]:

0.2207702619783588

2.1.3 b) Regression Tree

In [55]:

```
Features11 = PT[["GENDER","POSITION","EDUCATION"]].values
```

In [56]:

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()

encFeatures11 = enc.fit_transform(Features11).toarray()
encFeatures11 = pd.DataFrame(data=encFeatures11)
encFeatures11["AGE"] = list(PT["AGE"])
encFeatures11.head()
```

Out[56]:

	0	1	2	3	4	5	6	7	8	9	...	27	28	29	30	31	32	33	34	35
0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
2	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0

5 rows × 37 columns

In [57]:

```
TotalWages11 = PT[["TotalWages"]]
```

In [58]:

```
X_train11, X_test11, y_train11, y_test11 = train_test_split(encFeatures11, TotalWages11, random_sta
```

In [59]:

```
from sklearn.tree import DecisionTreeRegressor
Reg_tree = DecisionTreeRegressor()
```

In [60]:

```
Reg_tree.fit(X_train11,y_train11)
```

Out[60]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [61]:

```
Reg_tree.predict(X_test11)
```

Out[61]:

```
array([296.21428571, 231.58474576, 300.16578947, ..., 256.89864865,
       270.        , 248.33333333])
```

In [62]:

```
Reg_tree.score(X_test11,y_test11)
```

Out[62]:

```
0.07937481714160155
```

In [63]:

```
scorelist_regPTW =[]
length_regPTW = len(scorelist_regPTW)

#use for loop to list 10 scores calculated randomly by regression tree
for length_regFTW in range(1,11):
    from sklearn.tree import DecisionTreeRegressor
    Reg_tree_PTW = DecisionTreeRegressor()
    Reg_tree_PTW.fit(X_train11,y_train11)

    scorelist_regPTW.append(Reg_tree_PTW.score(X_test11,y_test11))
```

In [64]:

```
scorelist_regPTW
```

Out[64]:

```
[0.07934631995016694,
 0.08669430852120619,
 0.07825914764209674,
 0.07901581830750459,
 0.0788064435575816,
 0.07667102083053812,
 0.0778569167269577I,
 0.07744153430913048,
 0.07903573359430072,
 0.07897350635520428]
```

In [65]:

```
#Compute average score
np.mean(scorelist_regPTW )
```

Out[65]:

```
0.07921007497946873
```

For Part-Time TotalWages, we should use linear regression model as the score of linear regression (0.22077) is higher than the average score of regression tree (0.08243)

When we have information of gender, age, position, and education, we can use multivariable linear regression model to predict wages for part-time employees.

For conclusion, we create a table for complete comparison of each Model for Numerical Data:

In [66]:

```
scoreTable = {'Linear Regression': [0.35603,0.22077],
'Regression Tree': [0.03236,0.08243]}
scoreTable = pd.DataFrame(scoreTable)
scoreTable["Prediction"] = list(["Full-Time TotalWages","Part-Time TotalWages"])
scoreTable = scoreTable.set_index("Prediction")
scoreTable.T
```

Out[66]:

Prediction	Full-Time TotalWages	Part-Time TotalWages
Linear Regression	0.35603	0.22077
Regression Tree	0.03236	0.08243

2.2 Prediction for Categorical Data

There are 4 categorical data we can predict: Full-time Education, Full-time Position, Part-time Education, Part-time Position. Since the codes of Full-time Education and Part-time Education are almost the same, as well as the codes of Full-time Position and Part-time Position are almost the same, we just chose Full-time Education and Part-time Position as an illustration.

2.2.1 Full-Time Education

2.2.1 a) Logistic Regression

In [67]:

```
from sklearn.preprocessing import OneHotEncoder    # Use this one to transfer categorical data into
```

In [68]:

```
Features2 = FT[["GENDER","POSITION"]].values
Features2
```

Out[68]:

```
array([['male', 'Administrative'],
       ['female', 'Administrative'],
       ['female', 'Administrative'],
       ...,
       ['male', 'Transportation'],
       ['female', 'Transportation'],
       ['female', 'Transportation']], dtype=object)
```

In [69]:

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()

encFeatures2 = enc.fit_transform(Features2).toarray()
encFeatures2 = pd.DataFrame(data=encFeatures2)
encFeatures2["AGE"] = list(FT["AGE"])
encFeatures2["TotalWages"] = list(FT["TotalWages"])
encFeatures2.head()
```

Out[69]:

	0	1	2	3	4	5	6	7	8	9	...	21	22	23	24	25	26	27	28	AGE
0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	40
1	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	31
2	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	31
3	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	26
4	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	37

5 rows × 31 columns

In [70]:

```
X_train2, X_test2, y_train2, y_test2 = train_test_split(encFeatures2,FT["EDUCATION"], random_state=
```

In [71]:

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
```

In [72]:

```
log_reg.fit(X_train2, y_train2)
```

```
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
```

Out[72]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [73]:

```
X_test2.head()
```

Out[73]:

	0	1	2	3	4	5	6	7	8	9	...	21	22	23	24	25	26	27	28	A
125	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
228	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2335	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
726	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1113	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 31 columns

In [74]:

```
log_reg.predict(X_test2)
```

Out[74]:

```
array(['HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'College', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
```

In [75]:

```
log_reg.score(X_test2, y_test2)
```

Out[75]:

0.8937728937728938

In [76]:

```
scorelist_logregFTE =[]
length_logregFTE = len(scorelist_logregFTE)

#use for loop to list 10 scores calculated randomly by logistic regression
for length_logregFTE in range(1,11):
    from sklearn.linear_model import LogisticRegression
    log_regFTE = LogisticRegression(solver="lbfgs")
    log_regFTE.fit(X_train2,y_train2)

    scorelist_logregFTE.append(log_regFTE.score(X_test2,y_test2))
```

```
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWar
ning: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_
class option to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:758: Convergen
ceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:758: Convergen
ceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:758: Convergen
ceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:758: Convergen
ceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:758: Convergen
ceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
```

In [77]:

```
scorelist_logregFTE
```

Out[77]:

```
[0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938,
 0.8937728937728938]
```

In [78]:

```
#Compute average score
np.mean(scorelist_logregFTE)
```

Out[78]:

```
0.8937728937728938
```

Although there is no difference shown above , there is variation in score when we increase the run times in for loop. Therefore, for more accurate score, we can increase run time to at least 100. (Here is just for saving computation time)

2.2.1 b) K-Nearest Neighbors

In [79]:

```
from sklearn import neighbors
```

In [80]:

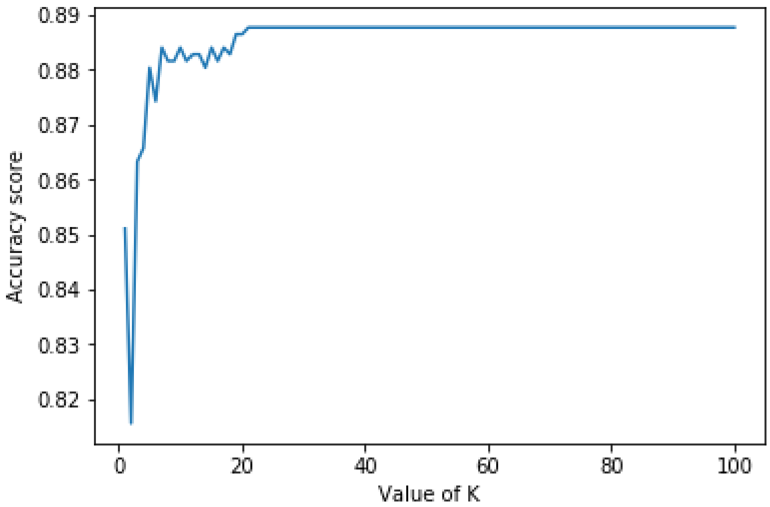
```
# We aim to select optimal K value that yields the highest accuracy
# Assume the range of k is from 1 to 100 in integer
# Returns a sequence of integers for k from 1 to 100
k_FTE_range = range(1,101)
scoresFTE_list=[]

for k in k_FTE_range:
    knnFTE = neighbors.KNeighborsClassifier(n_neighbors=k)
    knnFTE.fit(X_train2, y_train2)

    scoresFTE_list.append(knnFTE.score(X_test2, y_test2))
```

In [81]:

```
#Make a plot showing accuracy scores vs their corresponding k value
plt.plot(k_FTE_range, scoresFTE_list)
plt.xlabel('Value of K')
plt.ylabel('Accuracy score')
```

Out[81]:

```
Text(0, 0.5, 'Accuracy score')
```



In [82]:

```
#Indexing the point with highest accuracy score, which also has the optimal k value selection
Accuracy_highest_FTE = max(scoresFTE_list)
K_HighestAccuracy_FTE = k_FTE_range[scoresFTE_list.index(Accuracy_highest_FTE)]
print ("Optimal K is",K_HighestAccuracy_FTE, "with highest accuracy =",Accuracy_highest_FTE)
```

```
Optimal K is 21 with highest accuracy = 0.8876678876678876
```

In [83]:

```
from sklearn import neighbors

knnFTE = neighbors.KNeighborsClassifier(n_neighbors=21)
knnFTE.fit(X_train2,y_train2)
knnFTE.predict(X_test2)
knnFTE.score(X_test2,y_test2)
```

Out[83]:

```
0.8876678876678876
```

In [84]:

```
scorelistFTE =[]
lengthFTE = len(scorelistFTE)

for lengthFTE in range(1,11):
    from sklearn import neighbors
    knnFTE = neighbors.KNeighborsClassifier(n_neighbors=21)
    knnFTE.fit(X_train2,y_train2)

    scorelistFTE.append(knnFTE.score(X_test2,y_test2))
```

In [85]:

```
scorelistFTE
```

Out[85]:

```
[0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876,
 0.8876678876678876]
```

In [86]:

```
np.mean(scorelistFTE)
```

Out[86]:

```
0.8876678876678877
```

Although there is no difference shown above , there is variation in score when we increase the run times in for loop. Therefore, for more accurate score, we can increase run time to at least 100. (Here is just for saving computation time)

In some conditions, this score is high enough. However, if we want to let this model more accuracy, we can also normalize our data as z-score to manipulate.

2.2.1 c) Classification Tree

In [87]:

```
from sklearn.tree import DecisionTreeClassifier
Cat_tree_FTE = DecisionTreeClassifier()
```

In [88]:

```
Cat_tree_FTE.fit(X_train2,y_train2)
```

Out[88]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [89]:

```
Cat_tree_FTE.predict(X_test2)
```

Out[89]:

```
array(['HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'College', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'Some College', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'Post Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'College', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'College', 'HS Graduate', 'HS Graduate',
       'Some College', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'College', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
```

In [90]:

```
Cat_tree_FTE.score(X_test2,y_test2)
```

Out[90]:

```
0.8302808302808303
```

In [91]:

```
scorelist_catFTE =[]
length_catFTE = len(scorelist_catFTE)

#use for loop to list 10 scores calculated randomly by categorical tree
for length_catFTE in range(1,11):
    from sklearn.tree import DecisionTreeClassifier
    Cat_tree_FTE = DecisionTreeClassifier()
    Cat_tree_FTE.fit(X_train2,y_train2)

    scorelist_catFTE.append(Cat_tree_FTE.score(X_test2,y_test2))
```

In [92]:

```
scorelist_catFTE
```

Out[92]:

```
[0.8302808302808303,
 0.8376068376068376,
 0.8290598290598291,
 0.8290598290598291,
 0.8315018315018315,
 0.8315018315018315,
 0.8327228327228328,
 0.8302808302808303,
 0.833943833943834,
 0.8290598290598291]
```

In [93]:

```
np.mean(scorelist_catFTE)
```

Out[93]:

```
0.8315018315018315
```

2.2.1 d) Random Forest

In [94]:

```
from sklearn.ensemble import RandomForestClassifier
randomforest_FTE = RandomForestClassifier()
```

In [95]:

```
randomforest_FTE.fit(X_train2,y_train2)
```

```
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[95]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [96]:

```
randomforest_FTE.predict(X_test2)
```

Out[96]:

```
array(['HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'College', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
       'HS Graduate', 'HS Graduate', 'HS Graduate', 'HS Graduate',
```

In [97]:

```
randomforest_FTE.score(X_test2,y_test2)
```

Out[97]:

```
0.8766788766788767
```

In [98]:

```
scorelist_ranFTE =[]
length_ranFTE = len(scorelist_ranFTE)

#use for loop to list 10 scores calculated randomly by randomforest
for length_ranFTE in range(1,11):
    from sklearn.ensemble import RandomForestClassifier
    randomforest_FTE = RandomForestClassifier()
    randomforest_FTE.fit(X_train2,y_train2)

    scorelist_ranFTE.append(randomforest_FTE.score(X_test2,y_test2))
```

```
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [99]:

```
scorelist_ranFTE
```

Out[99]:

```
[0.873015873015873,
 0.8717948717948718,
 0.8693528693528694,
 0.8669108669108669,
 0.8742368742368742,
 0.8681318681318682,
 0.8705738705738706,
 0.8681318681318682,
 0.8717948717948718,
 0.8717948717948718]
```

In [100]:

```
np.mean(scorelist_ranFTE)
```

Out[100]:

```
0.8705738705738707
```

In [101]:

```
#Create a table for score comparison of different model of full time education
FscoreTable = {'Logistic Regression': [0.89377],
'K Nearest Neighbors': [0.88767],
'Classification Tree': [0.83040],
'Random Forest': [0.87082]}
FscoreTable = pd.DataFrame(FscoreTable)
FscoreTable["Prediction"] = list(["Full-Time Education"])
FscoreTable = FscoreTable.set_index("Prediction")
FscoreTable = FscoreTable.T
FscoreTable
```

Out[101]:

Prediction	Full-Time Education
Logistic Regression	0.89377
K Nearest Neighbors	0.88767
Classification Tree	0.83040
Random Forest	0.87082

As we can see from above table, for Full-Time Education, we should choose Logistic Regression as the score of this model is the highest, compared to other three ones

Score Rank for Full Time Education: Logisctic Regression > KNN > Random Forest > Classification Tree

When we have information of gender, age, position, and total wages, we can use logistic regression model to predict education for full-time employees.

2.2.2 Part-Time Position

2.2.2 a) Logistic Regression

In [102]:

```
Features22 = PT[["GENDER","EDUCATION"]].values
Features22
```

Out[102]:

```
array([['male', 'HS Graduate'],
       ['male', 'HS Graduate'],
       ['male', 'HS Graduate'],
       ...,
       ['female', 'HS Graduate'],
       ['female', 'HS Graduate'],
       ['male', 'HS Graduate']], dtype=object)
```

In [103]:

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()

encFeatures22 = enc.fit_transform(Features22).toarray()
encFeatures22 = pd.DataFrame(data=encFeatures22)
encFeatures22["AGE"] = list(PT["AGE"])
encFeatures22["TotalWages"] = list(PT["TotalWages"])
encFeatures22.head()
```

Out[103]:

	0	1	2	3	4	5	6	7	8	9	AGE	TotalWages
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	29	468.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	50	262.5
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	50	560.0
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	50	262.5
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	26	577.5

In [104]:

```
X_train22, X_test22, y_train22, y_test22 = train_test_split(encFeatures22,PT["POSITION"], random_st
```

In [105]:

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
```

In [106]:

```
log_reg.fit(X_train22, y_train22)
```

```
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
```

Out[106]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [107]:

```
X_test22.head()
```

Out[107]:

	0	1	2	3	4	5	6	7	8	9	AGE	TotalWages
9864	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	48	270.0
10584	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	29	300.0
14161	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	33	280.0
11206	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	21	176.0
3985	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	37	160.0

In [108]:

```
log_reg.predict(X_test22)
```

Out[108]:

```
array(['Healthcare', 'Food Service', 'Food Service', ..., 'Retail',
       'Janitorial Services', 'Healthcare'], dtype=object)
```

In [109]:

```
log_reg.score(X_test22, y_test22)
```

Out[109]:

0.23182474723158403

In [110]:

```
scorelist_logregPTP =[]
length_logregPTP = len(scorelist_logregPTP)

#use for loop to list 10 scores calculated randomly by logistic regression
for length_logregPTP in range(1,11):
    from sklearn.linear_model import LogisticRegression
    log_regPTP = LogisticRegression()
    log_regPTP.fit(X_train22,y_train22)

    scorelist_logregPTP.append(log_regPTP.score(X_test22, y_test22))
```

```
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
```

```
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
D:\Python NYU\lib\site-packages\sklearn\linear_model\logistic.py:460: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
```

In [111]:

```
scorelist_logregPTP
```

Out[111]:

```
[0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403,
 0.23182474723158403]
```

In [112]:

```
np.mean(scorelist_logregPTP)
```

Out[112]:

0.23182474723158406

Although there is no difference shown above , there is variation in score when we increase the run times in for loop. Therefore, for more accurate score, we can increase run time to at least 100. (Here is just for saving computation time)

2.2.2 b) K-Nearest Neighbors

In [113]:

```
from sklearn import neighbors
```

In [114]:

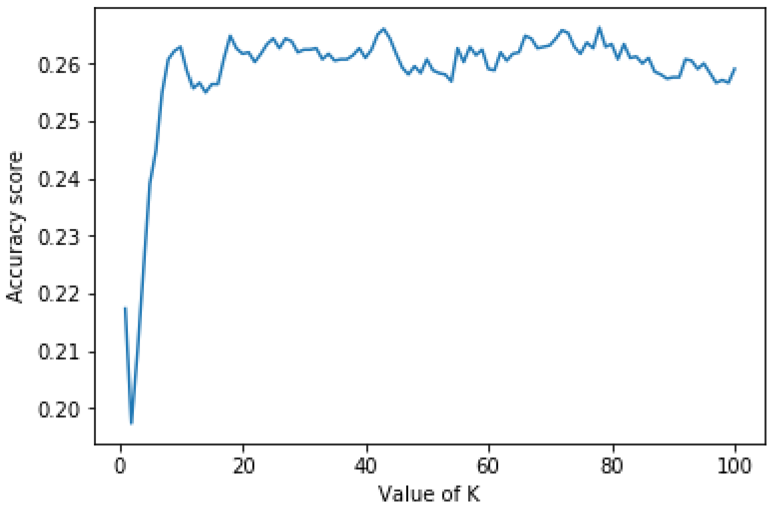
```
# We aim to select optimal K value that yields the highest accuracy
# Assume the range of k is from 1 to 100 in integer
# Returns a sequence of integers for k from 1 to 100
k_PTP_range = range(1,101)
scoresPTP_list=[]

#Here we use the same splitted data from Logistic Regression Part Time Position Part
for k in k_PTP_range:
    knnPTP = neighbors.KNeighborsClassifier(n_neighbors=k)
    knnPTP.fit(X_train22, y_train22)

    scoresPTP_list.append(knnPTP.score(X_test22, y_test22))
```

In [115]:

```
#Make a plot showing accuracy scores vs their corresponding k value
plt.plot(k_PTP_range, scoresPTP_list)
plt.xlabel('Value of K')
plt.ylabel('Accuracy score')
```

Out[115]:

```
Text(0, 0.5, 'Accuracy score')
```



In [116]:

```
#Select optimal k value
Accuracy_highest_PTP = max(scoresPTP_list)
K_HighestAccuracy_PTP = k_PTP_range[scoresPTP_list.index(Accuracy_highest_PTP)]
print ("Optimal K is",K_HighestAccuracy_PTP, "with highest accuracy =",Accuracy_highest_PTP)
```

```
Optimal K is 78 with highest accuracy = 0.266249398170438l3
```

In [117]:

```
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier(n_neighbors=78)
```

In [118]:

```
knn.fit(X_train22, y_train22)
```

Out[118]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=78, p=2,
           weights='uniform')
```

In [119]:

```
knn.predict(X_test22)
```

Out[119]:

```
array(['Food Service', 'Food Service', 'Food Service', ...,
       'Food Service', 'Food Service', 'Retail'], dtype=object)
```

In [120]:

```
knn.score(X_test22,y_test22)
```

Out[120]:

```
0.26624939817043813
```

In [121]:

```
scorelistPTP =[]
lengthPTP = len(scorelistPTP)

#Use a for loop to list 10 scores calculated randomly by knn
for lengthPTP in range(1,11):
    from sklearn import neighbors
    knnPTP = neighbors.KNeighborsClassifier(n_neighbors=78)
    knnPTP.fit(X_train22,y_train22)

    scorelistPTP.append(knnPTP.score(X_test22,y_test22))
```

In [122]:

```
scorelistPTP
```

Out[122]:

```
[0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813,
 0.26624939817043813]
```

In [123]:

```
#Compute average score
np.mean(scorelistPTP)
```

Out[123]:

```
0.26624939817043813
```

Although there is no difference shown above , there is variation in score when we increase the run times in for loop. Therefore, for more accurate score, we can increase run time to at least 100. (Here is just for saving computation time)

In some conditions, this score is high enough. However, if we want to let this model more accuracy, we can also normalize our data as z-score to manipulate.

2.2.2 c) Classification Tree

In [124]:

```
from sklearn.tree import DecisionTreeClassifier
Cat_tree = DecisionTreeClassifier()
```

In [125]:

```
Cat_tree.fit(X_train22,y_train22)
```

Out[125]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [126]:

```
Cat_tree.predict(X_test22)
```

Out[126]:

```
array(['Leisure And Hospitality', 'Other', 'Retail', ..., 'Food Service',
       'Janitorial Services', 'Telemarketing'], dtype=object)
```

In [127]:

```
Cat_tree.score(X_test22,y_test22)
```

Out[127]:

0.2676937891189215

In [128]:

```
scorelist_catPTP =[]
length_catPTP = len(scorelist_catPTP)

#use for loop to list 10 scores calculated randomly by categorical tree
for length_catPTP in range(1,11):
    from sklearn.tree import DecisionTreeClassifier
    Cat_tree_PTP = DecisionTreeClassifier()
    Cat_tree_PTP.fit(X_train22,y_train22)

    scorelist_catPTP.append(Cat_tree_PTP.score(X_test22,y_test22))
```

In [129]:

```
scorelist_catPTP
```

Out[129]:

```
[0.2698603755416466,
 0.268175252768416,
 0.26937891189215213,
 0.2691381800674049,
 0.2696196437168994,
 0.2667308618199326,
 0.26937891189215213,
 0.2674530572941743,
 0.2676937891189215,
 0.26841598459316324]
```

In [130]:

```
np.mean(scorelist_catPTP)
```

Out[130]:

0.2685844968704863

2.2.2 d) Random Forest

In [131]:

```
from sklearn.ensemble import RandomForestClassifier
randomforest = RandomForestClassifier()
```

In [132]:

```
randomforest.fit(X_train22,y_train22)
```

```
D:\Python NYU\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The d
efault value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[132]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [133]:

```
randomforest.predict(X_test22)
```

Out[133]:

```
array(['Leisure And Hospitality', 'Other', 'Retail', ..., 'Food Service',
       'Telemarketing', 'Telemarketing'], dtype=object)
```

In [134]:

```
randomforest.score(X_test22,y_test22)
```

Out[134]:

```
0.2563793933558016
```

In [135]:

```
scorelist_ranPTP =[]
length_ranPTP = len(scorelist_ranPTP)

#use for loop to list 10 scores calculated randomly by randomforest
for length_ranPTP in range(1,11):
    from sklearn.ensemble import RandomForestClassifier
    randomforest_PTP = RandomForestClassifier(n_estimators = 10)
    randomforest_PTP.fit(X_train22,y_train22)

    scorelist_ranPTP.append(randomforest_PTP.score(X_test22,y_test22))
```

In [136]:

```
scorelist_ranPTP
```

Out[136]:

```
[0.25252768415984594,
 0.25589792970630715,
 0.25493500240731826,
 0.2623976889744824,
 0.2522869523350987,
 0.2537313432835821,
 0.2590274434280212,
 0.25180548868560426,
 0.2616754935002407,
 0.2621569571497352]
```

In [137]:

```
#Compute average score
np.mean(scorelist_ranPTP)
```

Out[137]:

```
0.25664419836302355
```

In [138]:

```
#Create a table for score comparison of different model of part time position
PscoreTable = {'Logistic Regression': [0.23279],
'K Nearest Neighbors': [0.26625],
'Classification Tree': [0.26844],
'Random Forest': [0.25867]}
PscoreTable = pd.DataFrame(PscoreTable)
PscoreTable["Prediction"] = list(["Part-Time Position"])
PscoreTable = PscoreTable.set_index("Prediction")
PscoreTable = PscoreTable.T
PscoreTable
```

Out[138]:

Prediction	Part-Time Position
Logistic Regression	0.23279
K Nearest Neighbors	0.26625
Classification Tree	0.26844
Random Forest	0.25867

As we can see from above table, for Part-Time Position, we should choose Classification Tree as the score of this model is the highest, compared to other three ones

Score Rank for Part Time Position: Classfication Tree > KNN > Random Forest > Logistic Regression

When we have information of gender, age, education, and total wages, we can use logistic regression model to predict position for part-time employees.

3. Overall Conclusion

3.1 Numerical Data - Regression

For Full-time Total Wages, we should use Linear Regression model as the score of Linear Regression (0.35603) is higher than the average score of Regression Tree (0.03236). When we have information of gender, age, position, and education, we can use Multivariable Linear Regression model to predict wages for full-time employees.

For Part-Time TotalWages, we should use Linear Regression model as the score of Linear Regression (0.22077) is higher than the average score of Regression Tree (0.08243). When we have information of gender, age, position, and education, we can use Multivariable Linear Regression model to predict wages for part-time employees.

For conclusion, we create a table for complete comparison of each Model for Numerical Data:

In [139]:

```
scoreTable = {'Linear Regression': [0.35603,0.22077],
'Regression Tree': [0.03236,0.08243]}
scoreTable = pd.DataFrame(scoreTable)
scoreTable["Prediction"] = list(["Full-Time TotalWages","Part-Time TotalWages"])
scoreTable = scoreTable.set_index("Prediction")
scoreTable = scoreTable.T
scoreTable
```

Out[139]:

Prediction	Full-Time TotalWages	Part-Time TotalWages
Linear Regression	0.35603	0.22077
Regression Tree	0.03236	0.08243

3.2 Categorical Data - Classification

For Full-Time Education, we should choose Logistic Regression as the score of this model is the highest, compared to other three ones.When we have information of gender, age, position, and total wages, we can use Logistic Regression model to predict education for full-time employees.

For Part-Time Position, we should choose Classification Tree as the score of this model is the highest, compared to other three ones.When we have information of gender, age, education, and total wages, we can use Classification Tree model to predict position for part-time employees.

For conclusion, we create a table for complete comparison of each Model for Categorical Data:

In [140]:

```
PscoreTable = pd.DataFrame(data=PscoreTable)
PscoreTable = PscoreTable.reset_index()
FscoreTable = pd.DataFrame(data=FscoreTable)
FscoreTable = FscoreTable.reset_index()
```

In [141]:

```
Table = PscoreTable.merge(FscoreTable,on='index',how='inner')
Table.set_index("index")
```

Out[141]:

Prediction	Part-Time Position	Full-Time Education
index		
Logistic Regression	0.23279	0.89377
K Nearest Neighbors	0.26625	0.88767
Classification Tree	0.26844	0.83040
Random Forest	0.25867	0.87082