Capstone Project: Salaries and Wages Prediction

Ella Huang

28th/May/2020

This file consists of four parts below aimed to walk you through the journey to predict salaries and wages against a specified occupation.

---

1. Introduction

At this part, we shed light on describing the problem and data to be analyzed.The example occupation is Human Resources Manager with based in New York City and it's a full-time job.

Problems: As an employer, how much should I pay for a specified occupation? and how do I know what I pay is reasonable or even competitive in the labor market?

Background: Jay is an employer who has the business size around 500 employees, his business focuses on semiconductor and electronic product manufacturing, his factories locate in the non-metro area, however, he would like to establish an office in New York City as the business grows. As an employer, Jay has two concerns about the recruitment of Human Resources Manager(HRM), 1. Labor cost 2.Competitive salaries and wages. Therefore , he'd like to know how much he should pay for this occupation , which is reasonable and maybe competitive in the labor market.

Data: Data is the foundation and the key to explore our questions, so its reliability is the priority, because reliability determines validity.The source of our data is from Occupational Employment Statistics Survey from US Bureau of Labor Statistics, and the website: www.bls.gov/oes for reference

---

2.Methodology

This part, we focus on 1. The tools to be used for data analysis and data visualization in Python. 2. The Data Science methods to be applied. Please be noted that in view of the specific case we will look into, machine learning techniques won't be applied, including algorithms like Classification(LinearRegression,Ridge,LogisticRegression,DecisionTreeClassifier,KNeighborsClassifier,Support Vector Machine etc) or Clusters(like AgglomerativeClustering,DBCAN,KMeans etc). Foursquare API is not applicable for this case either.

Tools for Data Analysis: Numpy, Pandas

Tools for Data Visualization: Matplotlib

Data Science Methods:

1. Wrangling Data
2. Exploring Data
3. Model Development
4. Model Evaluation and Refinement

First of all, we import the tools that we need, then we read the datasets into pandas dataframe and check the relevant information around the dataframe

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
!conda install -c anaconda xlrd --yes
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.
```

Let's check how many worksheets in the workbook and what they are.

In [3]:
```
xl=pd.ExcelFile('Occupational Employment Stats.xlsx')
xl.sheet_names
```

Out[3]:
```
['All May 2019 Data', 'Field Descriptions']
```

We can see two sheets and their names displayed as well. Now we open Field Descriptions to look at the description and understand the features of the datasets. And we also set the column width considering the length of texts maybe long.

In [4]:
```
pd.set_option('max_colwidth',None)
xl.parse('Field Descriptions')
```

Out[4]:

	May 2019 OES Estimates	Unnamed: 1	Unnamed: 2
0	NaN	NaN	NaN
1	Occupational Employment Statistics (OES) Survey	NaN	NaN
2	Bureau of Labor Statistics, Department of Labor	NaN	NaN
3	website: www.bls.gov/oes	NaN	NaN
4	email: oesinfo@bls.gov	NaN	NaN
5	NaN	NaN	NaN
6	Not all fields are available for every type of estimate	NaN	NaN
7	NaN	NaN	NaN
8	Field	Field Description	NaN
9	area	U.S. (99), state FIPS code, Metropolitan Statistical Area (MSA) or New England City and Town Area (NECTA) code, or OES-specific nonmetropolitan area code	NaN
10	area_title	Area name	NaN
11	area_type	Area type: 1= U.S.; 2= State; 3= U.S. Territory; 4= Metropolitan Statistical Area (MSA) or New England City and Town Area (NECTA); 6= Nonmetropolitan Area	NaN
12	naics	North American Industry Classification System (NAICS) code for the given industry	NaN
13	naics_title	North American Industry Classification System (NAICS) title for the given industry	NaN
14	i_group	Industry level. Indicates cross-industry or NAICS sector, 3-digit, 4-digit, 5-digit, or 6-digit industry. For industries that OES no longer publishes at the 4-digit NAICS level, the “4-digit” designation indicates the most detailed industry breakdown available: either a standard NAICS 3-digit industry or an OES-specific combination of 4-digit industries. Industries that OES has aggregated to the 3-digit NAICS level (for example, NAICS 327000) will appear twice, once with the “3-digit” and once with the “4-digit” designation.	NaN
15	own_code	Ownership type: 1= Federal Government; 2= State Government; 3= Local Government; 123= Federal, State, and Local Government; 235=Private, State, and Local Government; 35 = Private and Local Government; 5= Private; 57=Private, Local Government Gambling Establishments (Sector 71), and Local Government Casino Hotels (Sector 72); 58= Private plus State and Local Government Hospitals; 59= Private and Postal Service; 1235= Federal, State, and Local Government and Private Sector	NaN
16	occ_code	The 6-digit Standard Occupational Classification (SOC) code or OES-specific code for the occupation	NaN
17	occ_title	SOC title or OES-specific title for the occupation	NaN
18	o_group	SOC occupation level. For most occupations, this field indicates the standard SOC major, minor, broad, and detailed levels, in addition to all-occupations totals. For occupations that OES no longer publishes at the SOC detailed level, the “detailed” designation indicates the most detailed data available: either a standard SOC broad occupation or an OES-specific combination of detailed occupations. Occupations that OES has aggregated to the SOC broad occupation level will appear in the file twice, once with the “broad” and once with the “detailed” designation.	.
19	tot_emp	Estimated total employment rounded to the nearest 10 (excludes self-employed).	NaN
20	emp_prse	Percent relative standard error (PRSE) for the employment estimate. PRSE is a measure of sampling error, expressed as a percentage of the corresponding estimate. Sampling error occurs when values for a population are estimated from a sample survey of the population, rather than calculated from data for all members of the population. Estimates with lower PRSEs are typically more precise in the presence of sampling error.	NaN
21	jobs_1000	The number of jobs (employment) in the given occupation per 1,000 jobs in the given area. Only available for the state and MSA estimates; otherwise, this column is blank.	.
22	loc quotient	The location quotient represents the ratio of an occupation’s share of employment in a given area to that occupation’s share of employment in the U.S. as a whole. For example, an occupation that makes up 10 percent of employment in a specific metropolitan area compared with 2 percent of U.S. employment would have a location quotient of 5 for the area in question. Only available for the state, metropolitan area, and nonmetropolitan area estimates; otherwise, this column is blank.	NaN
23	pct_total	Percent of industry employment in the given occupation. Percents may not sum to 100 because the totals may include data for occupations that could not be published separately. Only available for the national industry estimates; otherwise, this column is blank.	.
24	h_mean	Mean hourly wage	NaN
25	a_mean	Mean annual wage	NaN
26	mean_prse	Percent relative standard error (PRSE) for the mean wage estimate. PRSE is a measure of sampling error, expressed as a percentage of the corresponding estimate. Sampling error occurs when values for a population are estimated from a sample survey of the population, rather than calculated from data for all members of the population. Estimates with lower PRSEs are typically more precise in the presence of sampling error.	NaN
27	h_pct10	Hourly 10th percentile wage	NaN
28	h_pct25	Hourly 25th percentile wage	NaN
29	h_median	Hourly median wage (or the 50th percentile)	NaN
30	h_pct75	Hourly 75th percentile wage	NaN
31	h_pct90	Hourly 90th percentile wage	NaN
32	a_pct10	Annual 10th percentile wage	NaN
33	a_pct25	Annual 25th percentile wage	NaN

	May 2019 OES Estimates	Unnamed: 1	Unnamed: 2
34	a_median	Annual median wage (or the 50th percentile)	NaN
35	a_pct75	Annual 75th percentile wage	NaN
36	a_pct90	Annual 90th percentile wage	NaN
37	annual	Contains "TRUE" if only annual wages are released. The OES program releases only annual wages for some occupations that typically work fewer than 2,080 hours per year, but are paid on an annual basis, such as teachers, pilots, and athletes.	.
38	hourly	Contains "TRUE" if only hourly wages are released. The OES program releases only hourly wages for some occupations that typically work fewer than 2,080 hours per year and are paid on an hourly basis, such as actors, dancers, and musicians and singers.	.
39	NaN	NaN	NaN
40	Notes:	NaN	NaN
41	* = indicates that a wage estimate is not available	NaN	NaN
42	** = indicates that an employment estimate is not available	NaN	NaN
43	# = indicates a wage equal to or greater than $100.00 per hour or$208,000 per year	NaN	NaN
44	NaN	NaN	NaN

Read data into pandas dataframe.Check the number of rows and columns and data types

In [5]:
```
occ=xl.parse('All May 2019 Data')
print(occ.shape)
occ[0:3]
```

```
(395647, 30)
```

Out[5]:

	area	area_title	area_type	naics	naics_title	i_group	own_code	occ_code	occ_title	o_group	...	h_median	h_pct75	h_pct90	a
0	99	U.S.	1	000000	Cross-industry	cross-industry	1235	11-0000	Management Occupations	major	...	50.8	74.16	#	
1	99	U.S.	1	000000	Cross-industry	cross-industry	1235	13-0000	Business and Financial Operations Occupations	major	...	33.57	45.61	60.6	
2	99	U.S.	1	000000	Cross-industry	cross-industry	1235	15-0000	Computer and Mathematical Occupations	major	...	42.47	57.47	73.08	

3 rows × 30 columns

There are 395,647 rows and 30 columns and we need to look

In [6]:
```
occ.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395647 entries, 0 to 395646
Data columns (total 30 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   area            395647 non-null  int64  
 1   area_title      395647 non-null  object 
 2   area_type       395647 non-null  int64  
 3   naics           395647 non-null  object 
 4   naics_title     395647 non-null  object 
 5   i_group         395647 non-null  object 
 6   own_code        395647 non-null  int64  
 7   occ_code        395647 non-null  object 
 8   occ_title       395647 non-null  object 
 9   o_group         395647 non-null  object 
 10  tot_emp         395647 non-null  object 
 11  emp_prse        395647 non-null  object 
 12  jobs_1000_orig  225176 non-null  object 
 13  loc_quotient    207966 non-null  float64
 14  pct_total       165003 non-null  object 
 15  h_mean          395647 non-null  object 
 16  a_mean          395647 non-null  object 
 17  mean_prse       395647 non-null  object 
 18  h_pct10         395647 non-null  object 
 19  h_pct25         395647 non-null  object 
 20  h_median        395647 non-null  object 
 21  h_pct75         395647 non-null  object 
 22  h_pct90         395647 non-null  object 
 23  a_pct10         395647 non-null  object 
 24  a_pct25         395647 non-null  object 
 25  a_median        395647 non-null  object 
 26  a_pct75         395647 non-null  object 
 27  a_pct90         395647 non-null  object 
 28  annual          15709 non-null   object 
 29  hourly          741 non-null     object 
dtypes: float64(1), int64(3), object(26)
memory usage: 90.6+ MB
```

Through the observation, we find that from column 10 to column 27 except column 13, the data type is not what we want,so we need to convert them into float or int.

2.1 Wrangling Data

In [7]:
```
occ.iloc[:,11]=occ.iloc[:,11].replace('**',0)
occ.iloc[:,11]=occ.iloc[:,11].astype(float).round(2)
```

In [8]:
```
occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))]=occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))].replace('**',0)
occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))]=occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))].replace('*',0)
occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))]=occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))].replace('#',0)
occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))]=occ.iloc[:,[12]+list(range(14,16))+[17]+list(range(18,23))].astype(float).round(2)
```

In [9]:
```
occ.iloc[:,[10]+[16]+list(range(23,28))]=occ.iloc[:,[10]+[16]+list(range(23,28))].replace('**',0)
occ.iloc[:,[10]+[16]+list(range(23,28))]=occ.iloc[:,[10]+[16]+list(range(23,28))].replace('*',0)
occ.iloc[:,[10]+[16]+list(range(23,28))]=occ.iloc[:,[10]+[16]+list(range(23,28))].replace('#',0)
occ.iloc[:,[10]+[16]+list(range(23,28))]=occ.iloc[:,[10]+[16]+list(range(23,28))].astype(int)
```

</p>Check the data information again to see if the conversion we want is done before exploration. </p>

```
In [10]: occ.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395647 entries, 0 to 395646
Data columns (total 30 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   area            395647 non-null  int64
 1   area_title      395647 non-null  object
 2   area_type       395647 non-null  int64
 3   naics           395647 non-null  object
 4   naics_title     395647 non-null  object
 5   i_group         395647 non-null  object
 6   own_code        395647 non-null  int64
 7   occ_code        395647 non-null  object
 8   occ_title       395647 non-null  object
 9   o_group         395647 non-null  object
 10  tot_emp         395647 non-null  int64
 11  emp_prse        395647 non-null  float64
 12  jobs_1000_orig  225176 non-null  float64
 13  loc_quotient    207966 non-null  float64
 14  pct_total       165003 non-null  float64
 15  h_mean          395647 non-null  float64
 16  a_mean          395647 non-null  int64
 17  mean_prse       395647 non-null  float64
 18  h_pct10         395647 non-null  float64
 19  h_pct25         395647 non-null  float64
 20  h_median        395647 non-null  float64
 21  h_pct75         395647 non-null  float64
 22  h_pct90         395647 non-null  float64
 23  a_pct10         395647 non-null  int64
 24  a_pct25         395647 non-null  int64
 25  a_median        395647 non-null  int64
 26  a_pct75         395647 non-null  int64
 27  a_pct90         395647 non-null  int64
 28  annual          15709 non-null   object
 29  hourly          741 non-null     object
dtypes: float64(11), int64(10), object(9)
memory usage: 90.6+ MB
```

2.2 Exploring Data

In view of the specified occupation (HRM) is general and the industry requirement is not so significant. The features we pick up focus on : area_title,occ_title,tot_emp,emp_prse and datasets associated with average hourly wages and average annually wages.

```
In [11]: occ=occ.iloc[:,[1]+[8]+list(range(10,12))+list(range(15,28))]
occ.head()
```

Out[11]:

	area_title	occ_title	tot_emp	emp_prse	h_mean	a_mean	mean_prse	h_pct10	h_pct25	h_median	h_pct75	h_pct90	a_pct10	a_p
0	U.S.	Management Occupations	8054120	0.2	58.88	122480	0.1	24.03	34.35	50.80	74.16	0.00	49990	7
1	U.S.	Business and Financial Operations Occupations	8183750	0.2	37.56	78130	0.2	18.76	25.06	33.57	45.61	60.60	39020	5
2	U.S.	Computer and Mathematical Occupations	4552880	0.4	45.08	93760	0.5	21.79	30.22	42.47	57.47	73.08	45320	6
3	U.S.	Architecture and Engineering Occupations	2592680	0.5	42.69	88800	0.3	21.77	29.28	39.15	52.87	68.56	45280	6
4	U.S.	Life, Physical, and Social Science Occupations	1288920	0.7	37.28	77540	0.4	17.62	23.73	32.77	46.24	61.59	36640	4

Now at this stage, it's a moment to think about the job analysis and job description of HRM, as this requires the viewer to own the background knowledge and experience on recruitment. so we skip the process of job analysis and job description. But we'd like to emphasize that we conclude the skills required and job duties performed to weigh the job roles (s)he would play: Administration 10% , Compensation and Benefits 20%, Human Resource General 50%, Training and Development 20%

Combined with the location information: New York-Newark-Jersey City,we filter four series:

- ASM(Administrative Service Managers)
- CBM(Compensation and Benefits Managers)
- HRM(Human Resources Managers)
- TDM(Training and Development Managers)

In [12]:
```
k=['Administrative Services and Facilities Managers','Compensation and Benefits Managers',
   'Human Resources Managers','Training and Development Managers']
ASM=occ.loc[occ['occ_title']==k[0],occ.columns[[0]+list(range(2,occ.shape[1]))]]
ASM=ASM.groupby('area_title').mean().round(2)
ASM=ASM.loc['New York-Newark-Jersey City, NY-NJ-PA']
ASM
```

Out[12]:
```
tot_emp        25070.00
emp_prse           1.80
h_mean            67.69
a_mean        140800.00
mean_prse          0.90
h_pct10           37.09
h_pct25           48.35
h_median          61.67
h_pct75           79.07
h_pct90            0.00
a_pct10        77150.00
a_pct25       100570.00
a_median      128270.00
a_pct75       164460.00
a_pct90            0.00
Name: New York-Newark-Jersey City, NY-NJ-PA, dtype: float64
```

In [13]:
```
CBM=occ.loc[occ['occ_title']==k[1],occ.columns[[0]+list(range(2,occ.shape[1]))]]
CBM=CBM.groupby('area_title').mean().round(2)
CBM=CBM.loc['New York-Newark-Jersey City, NY-NJ-PA']
CBM
```

Out[13]:
```
tot_emp         1650.00
emp_prse           4.10
h_mean            87.39
a_mean        181770.00
mean_prse          2.80
h_pct10           52.10
h_pct25           63.61
h_median          78.74
h_pct75            0.00
h_pct90            0.00
a_pct10       108370.00
a_pct25       132320.00
a_median      163780.00
a_pct75            0.00
a_pct90            0.00
Name: New York-Newark-Jersey City, NY-NJ-PA, dtype: float64
```

In [14]:
```
HRM=occ.loc[occ['occ_title']==k[2],occ.columns[[0]+list(range(2,occ.shape[1]))]]
HRM=HRM.groupby('area_title').mean().round(2)
HRM=HRM.loc['New York-Newark-Jersey City, NY-NJ-PA']
HRM
```

Out[14]:
```
tot_emp        12020.00
emp_prse           2.40
h_mean            81.77
a_mean        170070.00
mean_prse          3.60
h_pct10           42.31
h_pct25           54.26
h_median          73.87
h_pct75           98.99
h_pct90            0.00
a_pct10        88000.00
a_pct25       112870.00
a_median      153650.00
a_pct75       205900.00
a_pct90            0.00
Name: New York-Newark-Jersey City, NY-NJ-PA, dtype: float64
```

In [15]:
```
TDM=occ.loc[occ['occ_title']==k[3],occ.columns[[0]+list(range(2,occ.shape[1]))]]
TDM=TDM.groupby('area_title').mean().round(2)
TDM=TDM.loc['New York-Newark-Jersey City, NY-NJ-PA']
TDM
```

Out[15]:
```
tot_emp         3230.00
emp_prse           3.60
h_mean            80.70
a_mean        167850.00
mean_prse          1.20
h_pct10           46.48
h_pct25           60.46
h_median          75.98
h_pct75           94.95
h_pct90            0.00
a_pct10        96690.00
a_pct25       125750.00
a_median      158050.00
a_pct75       197500.00
a_pct90            0.00
Name: New York-Newark-Jersey City, NY-NJ-PA, dtype: float64
```

2.3 Model Development

Then,it's time to concatenate the four series (ASM,CBM,HRM,TDM),and give it a name Merged, as we mentioned the weights each role should take, so we create a list named Weighted to store the weights.

Next, we create a column named Budgets to store the output of the model.

Our model is the sum of the each feature multiplied by its weights

In [16]:
```
m=[ASM,CBM,HRM,TDM]
Merged=pd.concat(m,axis=1)
Merged.columns=k
Merged['Budgets']=np.zeros(Merged.shape[0])
Weighted=[0.1,0.2,0.5,0.2]
a=[]
for x1,x2,x3,x4 in zip(Merged.iloc[:,0],Merged.iloc[:,1],Merged.iloc[:,2],Merged.iloc[:,3]):
    y=x1*Weighted[0]+x2*Weighted[1]+x3*Weighted[2]+x4*Weighted[3]
    a.append(y)
Merged['Budgets']=a
Merged
```

Out[16]:

	Administrative Services and Facilities Managers	Compensation and Benefits Managers	Human Resources Managers	Training and Development Managers	Budgets
tot_emp	25070.00	1650.00	12020.00	3230.00	9493.000
emp_prse	1.80	4.10	2.40	3.60	2.920
h_mean	67.69	87.39	81.77	80.70	81.272
a_mean	140800.00	181770.00	170070.00	167850.00	169039.000
mean_prse	0.90	2.80	3.60	1.20	2.690
h_pct10	37.09	52.10	42.31	46.48	44.580
h_pct25	48.35	63.61	54.26	60.46	56.779
h_median	61.67	78.74	73.87	75.98	74.046
h_pct75	79.07	0.00	98.99	94.95	76.392
h_pct90	0.00	0.00	0.00	0.00	0.000
a_pct10	77150.00	108370.00	88000.00	96690.00	92727.000
a_pct25	100570.00	132320.00	112870.00	125750.00	118106.000
a_median	128270.00	163780.00	153650.00	158050.00	154018.000
a_pct75	164460.00	0.00	205900.00	197500.00	158896.000
a_pct90	0.00	0.00	0.00	0.00	0.000

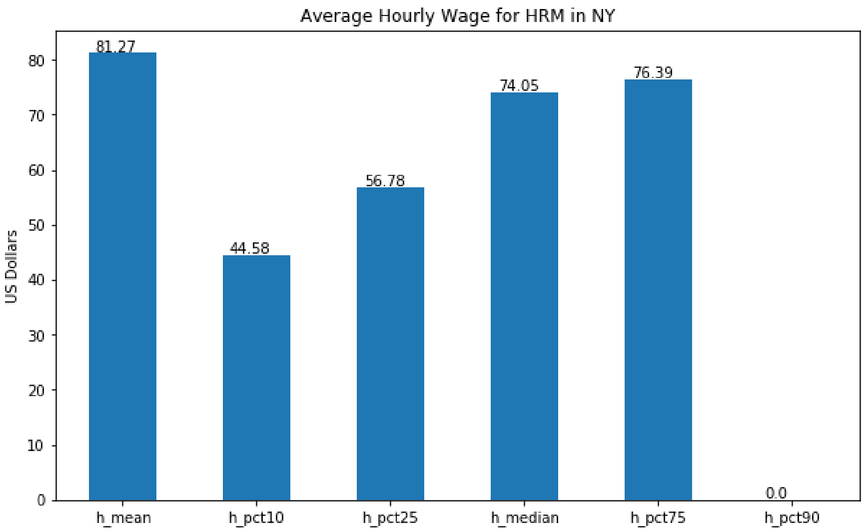
2.4 Model Evaluation and Refinement

Due to the model we build is to predict the salaries and wages for a specified occupation, and it's customized by cases , so we don't have uniformed data to evaluate and refine the model because the characteristics of each candidate is distinctive and unique. Therefore , this essential step is skipped for the model.

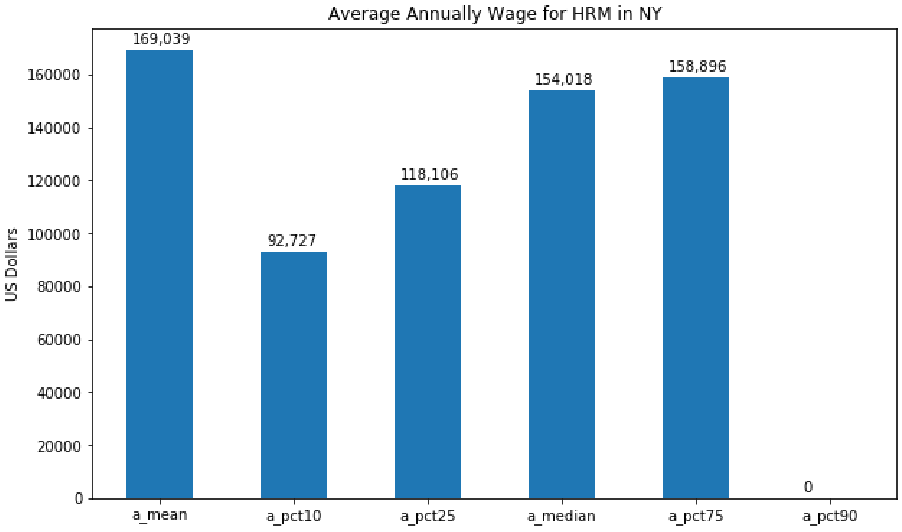
3. Results and Discussion

Now we can refer to the budgets for HRM, the mean of the hourly wage and the annual wage is **81.27 US dollars** and **169,039 US dollars** respectively, the median of the hourly wage and the annual wage is **74.046 US dollars** and **154,018 US dollars** respectively, also we can refer to the percentile of **10th, 25th, 50th, 75th** when an employer considers how much reasonably should be paid. However, we can see the 90th is 0 ,which means that's no data available to predict it. Besides salaries and wages prediction, benefits we should also take into account when design the remuneration packages, benefits mainly covers: Insurance, holidays, vacations, retirement plan and savings, leave plans, and legally required benefits. As for insurance , especially health insurance which mainly include medical care, vision care, dental care, outpatient prescription drugs. In a word, we should keep an eye on the various factors that affect compensation and benefits, take care of the balance between the labor costs and remuneration packages.

In [17]:
```
AHW=Merged.iloc[[2]+list(range(5,10)),4].round(2)
AHW.plot(kind='bar',figsize=(10,6))
plt.title('Average Hourly Wage for HRM in NY')
plt.ylabel('US Dollars')
plt.xticks(rotation=0)
for i,v in enumerate(AHW):
    plt.annotate(v,xy=[(i-0.2),(v+0.3)])
plt.show()
```



In [18]:
```
AAW=Merged.iloc[[3]+list(range(10,15)),4].astype(int)
AAW.plot(kind='bar',figsize=(10,6))
plt.title('Average Annually Wage for HRM in NY')
plt.ylabel('US Dollars')
plt.xticks(rotation=0)
for i,v in enumerate(AAW):
    plt.annotate('{:,}'.format(v),xy=[(i-0.2),(v+2500)])
plt.show()
```



4.Conclusion

From the data analysis above and the diagram shows, we can conclude the points below for reference:

- The estimated average hourly wage for HRM in New York-Newark-Jersey City, USA is **USD 81.27** and the median is **USD 74.05**
- The estimated average annually wage for HRM in New York-Newark-Jersey City, USA is **USD 169,039** and the median is **USD 154,018.**

Besides,please be noted that we don't take CPI (Consumer Price Index) into account.

Hope you enjoy this example and learn a bit of how to predict salaries and wages for a specified occupation.

In [ ]: