به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

هوش مصنوعی

گزارش تمرین کامپیوتری صفر

نام و نام خانوادگی:

فاطمه شاه حسینی

شماره دانشجویی:

810199440

هدف پروژه : پیش بینی نواقص یک دیتا با کمک تحلیل آماری

صورت پروژه : یک فایل داده train داریم که شامل اطلاعات مربوط به افراد متقاضی کار است و با توجه به شرایط سنی، تحصیلاتی، جنسیتی، نژادی، مالی، خانوادگی و سابقه کاری میزان دستمزد این افراد را مشخص کرده است.

در فایلی دیگر به نام test همین اطلاعات متعلق به افراد دیگری وجود دارد که ما باید با کمک تحلیل اماری فایل train و پیدا کردن ویژگی هایی که بیشترین تاثیر در میزان دستمزد را دارند، دستمزد این افراد را پیش بینی کنیم.

-1

خواندن از روی فایل csv و ایجاد یک دیتافریم به هدف انجام تحلیل اماری و پیدا کردن بهترین و موثرترین ویژگی در میزان دستمزد.

In [1]:
```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import time
from scipy.stats import norm
```

In [2]:
```
people_data = pd.read_csv("./train.csv")
```

بررسی متد های:

head(), : نمایش 5 عنصر اول دیتا فریم

In [4]:
```
people_data.head()
```

Out[4]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

tail() : نمایش 5 عنصر اخر دیتا فریم

```
In [5]: people_data.tail()
```

Out[5]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
29820	48	Local-gov	127921	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	40	United-States	<=50K
29821	32	Private	42617	Some-college	10	Divorced	Prof-specialty	Not-in-family	White	Female	0	0	30	United-States	<=50K
29822	47	Local-gov	191389	HS-grad	9	Divorced	Adm-clerical	Unmarried	White	Female	0	0	35	United-States	<=50K
29823	38	Private	187983	Prof-school	15	Married-civ-spouse	Sales	Wife	White	Female	0	0	40	United-States	<=50K
29824	18	Private	215110	HS-grad	9	Never-married	Handlers-cleaners	Own-child	Black	Male	0	0	40	United-States	<=50K

info(), : نمایش تایپ هر ستون و تعداد عناصر غیر تهی هر ستون به همراه حافظه مصرفی

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29825 entries, 0 to 29824
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             29825 non-null  int64
 1   workclass       28154 non-null  object
 2   fnlwgt          29825 non-null  int64
 3   education       29825 non-null  object
 4   education-num   29825 non-null  int64
 5   marital-status  29825 non-null  object
 6   occupation      28149 non-null  object
 7   relationship    29825 non-null  object
 8   race            29825 non-null  object
 9   sex             29825 non-null  object
 10  capital-gain    29825 non-null  int64
 11  capital-loss    29825 non-null  int64
 12  hours-per-week  29825 non-null  int64
 13  native-country  29288 non-null  object
 14  salary          29825 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.4+ MB
```

describe() : نمایش اطلاعات کلی درباره دیتا فریم شامل صدک های اول و دوم و سوم و چهارم، تعداد کل نمونه ها، کمترین و بیشترین مقدار، میانگین و انحراف معیار و ...

```
In [3]: people_data.describe()
```

Out[3]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	29825.000000	2.982500e+04	29825.000000	29825.000000	29825.000000	29825.000000
mean	38.581425	1.897389e+05	10.079229	1094.586052	86.476513	40.434334
std	13.652005	1.053738e+05	2.571678	7485.908646	401.195078	12.308174
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178490e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.785170e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.368790e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

-2

بررسی ستون های عددی و دسته ای:

با کمک ستون Dtype خروجی تابع info می توان فهمید ستون های age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week عددی و workclass, education, occupation, relationship, race, sex, native-country, salary دسته ای هستند.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29825 entries, 0 to 29824
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             29825 non-null  int64
 1   workclass       28154 non-null  object
 2   fnlwgt          29825 non-null  int64
 3   education       29825 non-null  object
 4   education-num   29825 non-null  int64
 5   marital-status  29825 non-null  object
 6   occupation      28149 non-null  object
 7   relationship    29825 non-null  object
 8   race            29825 non-null  object
 9   sex             29825 non-null  object
 10  capital-gain    29825 non-null  int64
 11  capital-loss    29825 non-null  int64
 12  hours-per-week  29825 non-null  int64
 13  native-country  29288 non-null  object
 14  salary          29825 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.4+ MB
```

ستون های دسته ای با کمک توابع زیر که مقادیر مختلف هر ویژگی را پیدا کرده و به جای هر کدام یک کد اختصاص می دهند. Label encoding

```
In [13]:
people_data['sex'] = people_data['sex'].astype('category')
people_data['sex'] = people_data['sex'].cat.codes

people_data['workclass'] = people_data['workclass'].astype('category')
people_data['workclass'] = people_data['workclass'].cat.codes

people_data['education'] = people_data['education'].astype('category')
people_data['education'] = people_data['education'].cat.codes

people_data['occupation'] = people_data['occupation'].astype('category')
people_data['occupation'] = people_data['occupation'].cat.codes

people_data['race'] = people_data['race'].astype('category')
people_data['race'] = people_data['race'].cat.codes

people_data['marital-status'] = people_data['marital-status'].astype('category')
people_data['marital-status'] = people_data['marital-status'].cat.codes

people_data['relationship'] = people_data['relationship'].astype('category')
people_data['relationship'] = people_data['relationship'].cat.codes

people_data['native-country'] = people_data['native-country'].astype('category')
people_data['native-country'] = people_data['native-country'].cat.codes


people_data.loc[590:600]
```

Out[13]:

Out[13]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
590	42	3	137390	11	9	2	9	0	4	1	0	0	40	38	<=50K
591	55	3	105138	11	9	2	9	5	1	0	0	0	40	38	<=50K
592	60	3	39352	5	4	4	13	1	4	1	0	0	48	38	>50K
593	31	3	168387	9	13	2	9	0	4	1	7688	0	40	1	>50K
594	23	3	117789	9	13	4	0	3	4	0	0	0	40	38	<=50K
595	27	3	267147	11	9	4	11	3	4	1	0	0	40	38	<=50K
596	23	3	99399	15	10	4	9	4	0	0	0	0	25	38	<=50K
597	42	5	214242	14	15	2	9	0	4	1	0	1902	50	38	>50K
598	25	3	200408	15	10	4	12	1	4	1	2174	0	40	38	<=50K
599	49	3	136455	9	13	4	9	1	4	0	0	0	45	38	<=50K
600	32	3	239824	9	13	4	12	1	4	1	0	0	40	38	<=50K

بخشی از دیتا فریم بعد از label encode شدن

-3

پیدا کردن ستون هایی که داده های تهی دارند و پر کردن آنها.

3-1 Find number of NaN values in each column

```
In [9]: people_data.isna().sum()
```

```
Out[9]: age                0
        workclass       1671
        fnlwgt             0
        education          0
        education-num      0
        marital-status     0
        occupation      1676
        relationship       0
        race               0
        sex                0
        capital-gain       0
        capital-loss       0
        hours-per-week     0
        native-country   537
        salary             0
        dtype: int64
```

تعداد nan های هر ستون

این داده های تهی را می توان در ستون های عددی با مقدار میانگین ستون و در ستون های دسته ای با mod ستون پر کرد.

3-2 Replace the NaN values with Mean for numerical colunms and with Mode for categorical columns.

```
In [10]: numeric_columns = people_data.select_dtypes(include=['number']).columns
people_data[numeric_columns] = people_data[numeric_columns].fillna(people_data.mean(numeric_only=True))

Categorical_columns = people_data.select_dtypes(exclude=['number']).columns.difference(['salary'])
befor_fill_nan = people_data[Categorical_columns]
people_data[Categorical_columns] = people_data[Categorical_columns].transform(lambda a: a.fillna(a.mode()[0]))
after_fill_nan = people_data[Categorical_columns]
```

فواید پر کردن خانه های خالی با میانگین: این کار برای تعداد داده کم جواب می دهد و باعث می شود مقدار میانگین کل ثابت بماند.

معایب پر کردن خانه های خالی با میانگین: در تعداد داده زیاد باعث می شود انحراف معیار زیاد شود و به تخمین خوبی نرسیم.

-4

حذف ستون هایی که مقادیر تکراری کمی دارند و برای یادگیری به ما کمک نمی کنند،

rate_unique_fnlwgt = 20337 / 29825 = 0.68

rate_unique_hours-per-week = 94 / 29825 = 0.003

rate_marital-statust = 7 / 29825 = 0.0002

rate_age = 72 / 29825 = 0.002

rate_unique_workclass = 9 / 29825 = 0.0002

rate_education = 16 / 29825 = 0.0004

rate_education-num = 16 / 29825 = 0.0004

rate_occupation = 15 / 29825 = 0.0004

rate_relationship = 6 / 29825 = 0.0002

rate_unique_native-country = 42 / 29825 = 0.001

```
In [34]: people_data['age'].unique().size / people_data['age'].size
Out[34]: 0.0024140821458507964
```

4- Drop columns that are unique and we can not use them for training: rate_unique_fnlwgt = 20337 / 29825 = 0.68 * rate_unique_hours-per-week = 94 / 29825 = 0.003 * rate_marital-statust = 7 / 29825 = 0.0002 * rate_age = 72 / 29825 = 0.002 * rate_unique_workclass = 9 / 29825 = 0.0002 * rate_education = 16 / 29825 = 0.0004 * rate_education-num = 16 / 29825 = 0.0004 * rate_occupation = 15 / 29825 = 0.0004 * rate_relationship = 6 / 29825 = 0.0002 * rate_unique_native-country = 42 / 29825 = 0.001

```
In [16]: people_data = people_data.drop(columns=['fnlwgt'])
people_data.head()
```

که درصد تنوع fnlwgt نسبت به بقیه خیلی زیاد است و کمک زیادی به تحلیل ما نمی کند.

-5

در این بخش به کمک Boolean mask ها شرایط هر کدام را مشخص کرده و با عبور کل دیتا از این mask ها سایز مجموعه دیتای خواست مسله را حساب می کنیم.

چند نفر مرد و زن داریم؟ 19956 تا مرد و 9869 زن (با استفاده از تابع value_counts که تعداد تکرار مقادیر مختلف هر ستون sex را می دهد)

چند نفر از مردان متاهلند؟ 12373

(Married-civ-spouse, Married-spouse-absent, Married-AF-spouse همه این موارد را متاهل حساب کردیم.)

5-1 Count nunber of men and women

```
In [17]:
# 1 are men and 0 are women
people_data['sex'].value_counts()
```

```
Out[17]:
1    19956
0     9869
Name: sex, dtype: int64
```

5-2 How many men are married?

```
In [18]:
people_data.rename(columns={"marital-status": "marital"}, inplace=True)
marital_cond = (people_data.marital == 1) | (people_data.marital == 2) | (people_data.marital == 3)
sex_cond = (people_data.sex == 1)
len(people_data[marital_cond & sex_cond])
```

```
Out[18]: 12373
```

-6

چند نفر سیاه پوست ، بالای سی سال، کار کردن انفرادی؟ 1373

```
In [19]:
race_cond = (people_data.race == 2)
age_cond = (people_data.age > 30)
workless_cond = (people_data.workclass == 3)

len(people_data[race_cond & age_cond & workless_cond])
```

```
Out[19]: 1373
```

```
In [20]:
len(np.where((people_data.race == 2) & (people_data.age > 30) & (people_data.workclass == 3) )[0])
```

```
Out[20]: 1373
```

اینجا به دو روش فیلتر کردن دیتا را انجام داده ایم. روش اول همان Boolean masking که بهتر و گویا تر است و در روش دوم از کتابخانه numpy و متد where استفاده کرده ایم که دیتا با شرایط داده شده را جدا می کند.

-7

میانگین ساعت کاری افرادی که لیسانسه هستند؟(با وکتوریزیشن)

7 Average working hours of people with a bachelor's degree (Vectorization)

In [36]:
```
people_data.rename(columns={"hours-per-week": "hoursPerWeek"}, inplace=True)
education_cond = (people_data.education == 9)
hours = people_data[education_cond].hoursPerWeek

startTime1 = time.time()
meanhours = hours.mean()
endTime1 = time.time()

print("Mean working hours : ", meanhours)
print("Execution time : ", endTime1 - startTime1)
```

```
Mean working hours :  42.546669366268475
Execution time :  0.0009958744049072266
```

-8

میانگین ساعت کاری افرادی که لیسانسه هستند؟(با حلقه)

8 Average working hours of people with a bachelor's degree (Loop)

In [37]:
```
startTime2 = time.time()
sum = 0
for h in hours:
    sum += h
meanhours = sum / hours.size
endTime2 = time.time()

print("Mean working hours : ", meanhours)
print("Execution time : ", endTime2 - startTime2)
```

```
Mean working hours :  42.546669366268475
Execution time :  0.0019958019256591797
```

زمان در وکتوریزیشن :

0.0009982585906982422

زمان در حلقه :

0.001994609832763672

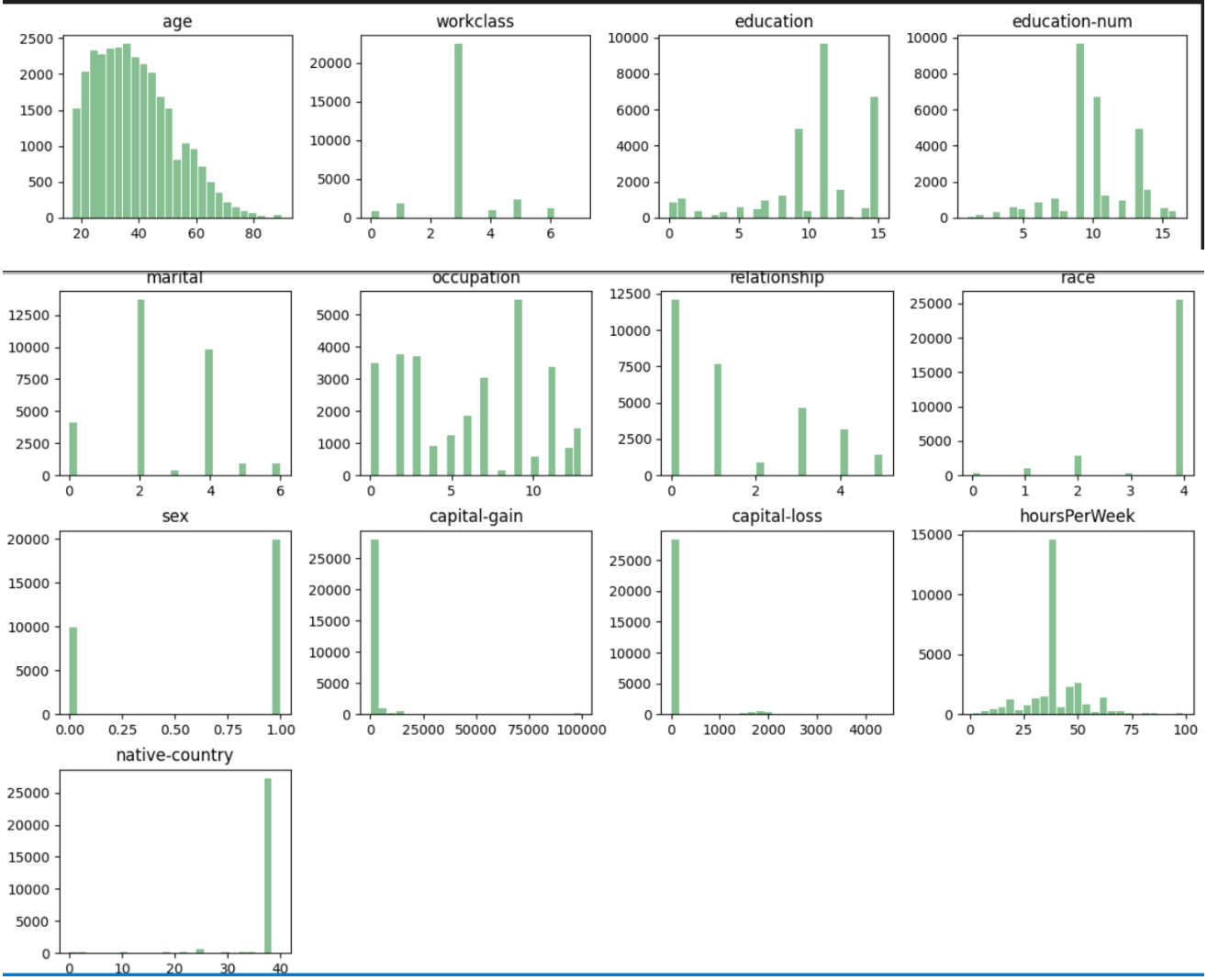
تقریبا در حلقه دوبرابر زمان لازم است چون در وکتوریزیشن دیتا ها به صورت موازی پراسس می شوند و زمان بسیار کمتری صرف می کنند.

-9

نمایش توزیع هر ستون از داده روی نمودار هیستوگرام

```
people_data.hist(bins=25, grid=False, figsize=(15,12), color='#86bf92', zorder=2, rwidth=0.9)
```

```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'workclass'}>,
        <AxesSubplot:title={'center':'education'}>,
        <AxesSubplot:title={'center':'education-num'}>],
       [<AxesSubplot:title={'center':'marital'}>,
        <AxesSubplot:title={'center':'occupation'}>,
        <AxesSubplot:title={'center':'relationship'}>,
        <AxesSubplot:title={'center':'race'}>],
       [<AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'capital-gain'}>,
        <AxesSubplot:title={'center':'capital-loss'}>,
        <AxesSubplot:title={'center':'hoursPerWeek'}>],
       [<AxesSubplot:title={'center':'native-country'}>, <AxesSubplot:>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



-10

نرمال سازی دیتا ها

از انجایی که میانگین گیری از کد هایی که به داده های غیر عددی نسبت دادیم معنا ندارد، ابتدا ستون های غیر عددی را حذف می کنیم،

```
In [24]:
numeric_people_data = people_data.drop(columns=['education', 'native-country', 'occupation', 'relationship','workclass',

normalized_people_data = (numeric_people_data - numeric_people_data.mean()) / numeric_people_data.std()

normalized_people_data
```

Out[24]:

	age	education-num	marital	race	sex	capital-gain	capital-loss	hoursPerWeek
0	0.030660	1.135745	0.918457	0.393610	0.703222	0.144193	-0.215547	-0.035288
1	0.836403	1.135745	-0.407149	0.393610	0.703222	-0.146220	-0.215547	-2.228952
2	-0.042589	-0.419659	-1.732754	0.393610	0.703222	-0.146220	-0.215547	-0.035288
3	1.056151	-1.197362	-0.407149	-1.962521	0.703222	-0.146220	-0.215547	-0.035288
4	-0.775082	1.135745	-0.407149	-1.962521	-1.421978	-0.146220	-0.215547	-0.035288
...	...	...	...	...	...	...	...	...
29820	0.689904	1.135745	-0.407149	0.393610	0.703222	-0.146220	-0.215547	-0.035288
29821	-0.482085	-0.030808	-1.732754	0.393610	-1.421978	-0.146220	-0.215547	-0.847756
29822	0.616655	-0.419659	-1.732754	0.393610	-1.421978	-0.146220	-0.215547	-0.441522
29823	-0.042589	1.913448	-0.407149	0.393610	-1.421978	-0.146220	-0.215547	-0.035288
29824	-1.507575	-0.419659	0.918457	-1.962521	0.703222	-0.146220	-0.215547	-0.035288

29825 rows × 8 columns

-11

محاسبه میانگین و انحراف معیار برای حالت دستمزد بیشتر از 50 و کمتر از 50.

به کمک تابع loc مواردی که دستمزد بیشتر و کمتر از 50 هزار دارند را از هم جدا می کنیم و به کمک توابع mean و std میانگین و انحراف معیار هر یک را به صورت vectorization حساب می کنیم.

11-1 claculate mean and std of people with more than 50K salary

```
:
people_with_more_than_50_salary = people_data.loc[people_data['salary'] == '>50K']

numeric_people_with_more_than_50_salary = people_with_more_than_50_salary.drop(columns=['salary'])

mean_people_with_more_than_50_salary = numeric_people_with_more_than_50_salary.mean()

std_people_with_more_than_50_salary = numeric_people_with_more_than_50_salary.std()
```

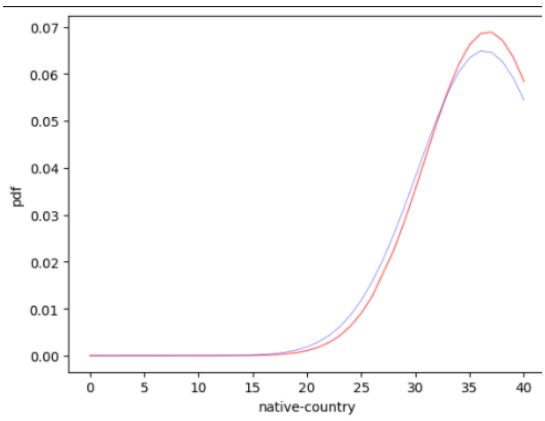
11-2 claculate mean and std of people with less than 50K salary

In [26]:
```
people_with_less_than_50_salary = people_data.loc[people_data['salary'] == '<=50K']

numeric_people_with_less_than_50_salary = people_with_less_than_50_salary.drop(columns=['salary'])

mean_people_with_less_than_50_salary = numeric_people_with_less_than_50_salary.mean()

std_people_with_less_than_50_salary = numeric_people_with_less_than_50_salary.std()
```

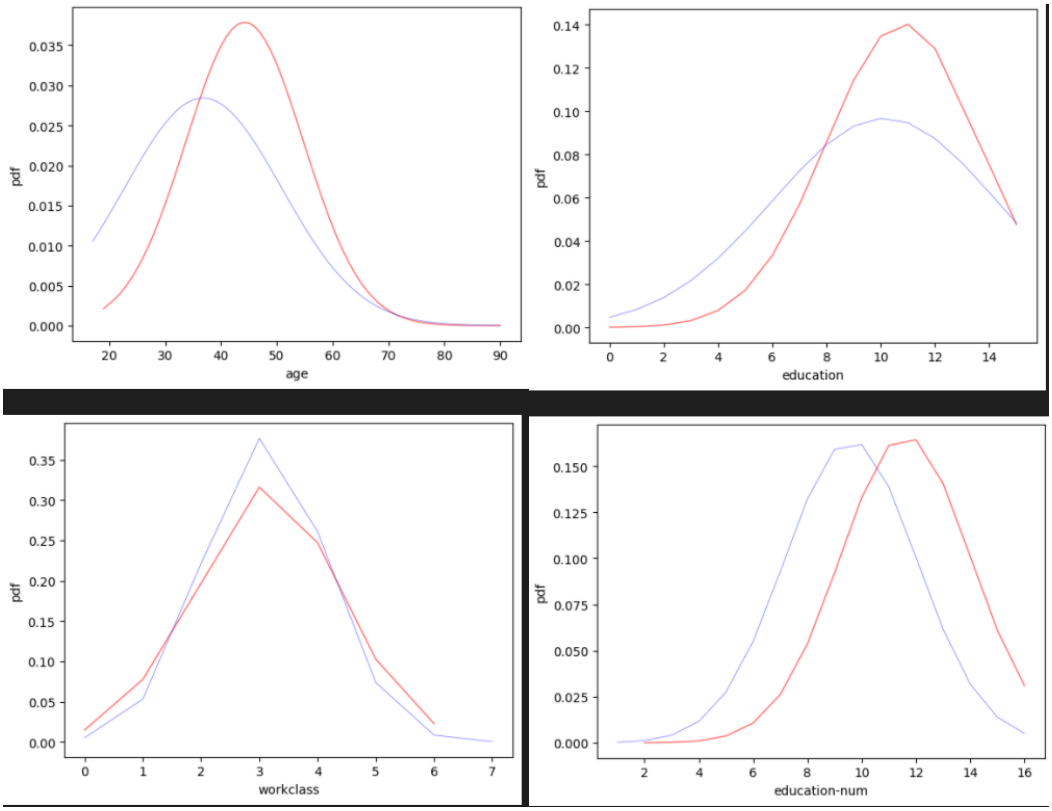
نمایش نمودار PDF مرتبط با هر ویژگی:

```
cols = numeric_people_with_more_than_50_salary.columns
for i in range(len(cols)):
    fig, ax = plt.subplots(1, 1)
    plt.xlabel(numeric_people_with_more_than_50_salary.columns[i])
    plt.ylabel("pdf")

    x = numeric_people_with_more_than_50_salary.iloc[:, i].sort_values()
    ax.plot(x, norm.pdf(x, mean_people_with_more_than_50_salary[i], std_people_with_more_than_50_salary[i]),'r-', lw=1, alpha

    x = numeric_people_with_less_than_50_salary.iloc[:, i].sort_values()
    ax.plot(x, norm.pdf(x, mean_people_with_less_than_50_salary[i], std_people_with_less_than_50_salary[i]),'b-', lw=0.5, alp
```

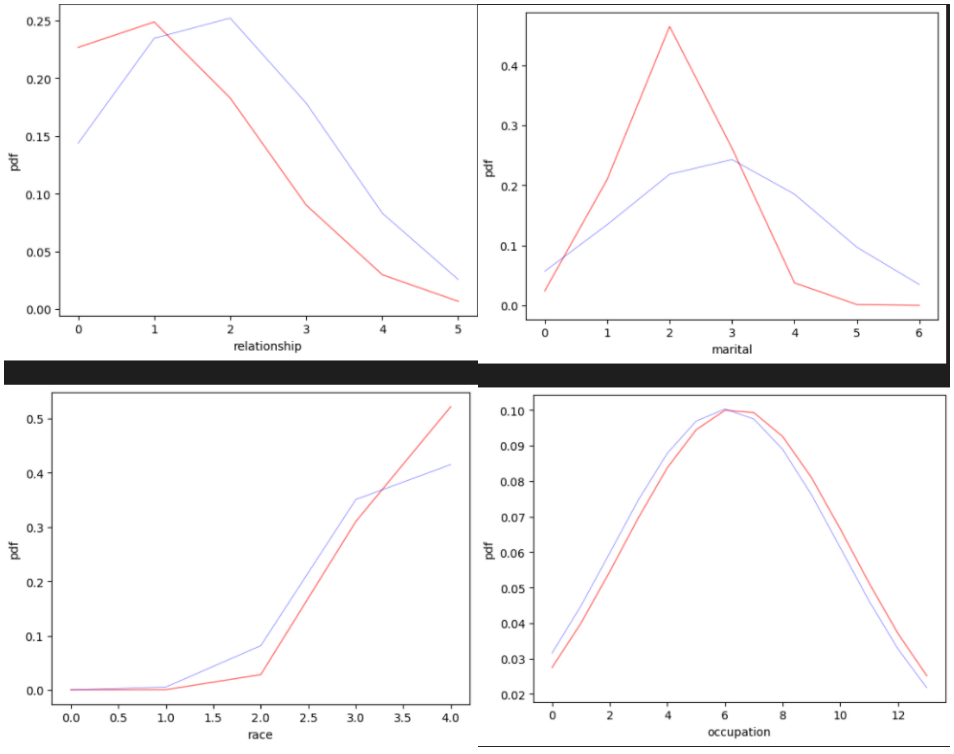
برای هر ستون که نشان دهنده یک ویژگی است نمودار pdf را رسم می کنیم. برای محاسبه تابع pdf از متد pdf در کتابخانه spicy.stat استفاده می کنیم که با گرفتن linespace, mean , std تابع پی دی اف را می سازد. داده ای که به عنوان ورودی به پی دی اف می دهیم یکبار داده های مرتبط با افراد با دستمزد بالاتر از 50 هزار و یکبار داده های مرتبط با افراد با دستمزد کمتر از 50 هزار است. و در نهایت هر دو را روی یک نمودار می کشیم.نمودار قرمز رنگ مرتبط با بالای 50 هزار و آبی زیر 50 هزار است.

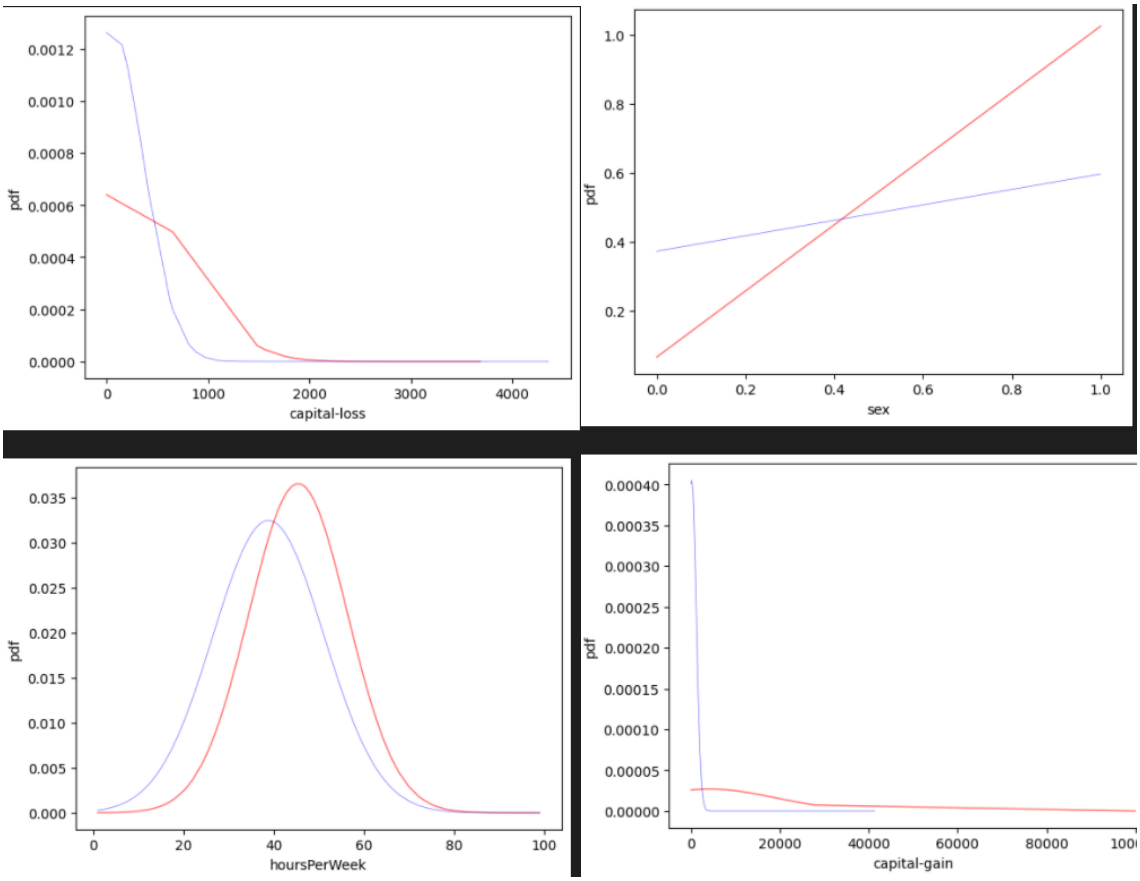


در این نمودار پیوستگی زیادی بین دو نمودار می بینیم که یعنی کشور مادری تاثیرچشمگیری در دستمزد ندارد و اگر مثلا کشور مکزیک به عدد 30، encode شده باشد، احتمال اینکه فردی که در مکزیک متولد شده بیشتر از 50 هزار یا کمتر از آن بگیرد برابر است.



در نمودار education تفاوت چشمگیر تری می بینیم، و به وضوح کسی که 10 کلاس درس خوانده است احتمال زیادی دارد که بیشتر از 50 هزار دستمزد بگیرد.





و در نمودار سود سرمایه ای capital gain می بینیم که اگر سود سرمایه ای کسی صفر است تقریبا صد در صد حقوق کمتر از 50 هزار دارد و در غیر این صورت حقوق بالاتری دارد. و به نظر می رسد که این ویژگی بیشتر از بقیه توانسته دو نمودار را از هم تفکیک کند که به معنای تاثیرگذاری بیشتر آن در مقدار دستمزد است.

حالا برای مقایسه دقیق تر هربار با توجه به یک ویژگی ، دیتای فایل test را پیش بینی و با واقعیت مقایسه می کنیم تا میزان صحت پیش بینی با هر ویژگی را بسنجیم.

ابتدا فایل تست را در یک دیتافریم جدید لود می کنیم و مثل قبل ستون های غیر عددی را به عددی encode می کنیم، ستون دستمزد واقعی را هم دریک لیست جدا برای مرحله محاسبه خطا ذخیره می کنیم.

In [31]:
```
test_data = pd.read_csv("./test.csv")
read_salaries = test_data['salary']
```

In [32]:
```
test_data['sex'] = test_data['sex'].astype('category')
test_data['sex'] = test_data['sex'].cat.codes

test_data['workclass'] = test_data['workclass'].astype('category')
test_data['workclass'] = test_data['workclass'].cat.codes

test_data['education'] = test_data['education'].astype('category')
test_data['education'] = test_data['education'].cat.codes

test_data['occupation'] = test_data['occupation'].astype('category')
test_data['occupation'] = test_data['occupation'].cat.codes

test_data['race'] = test_data['race'].astype('category')
test_data['race'] = test_data['race'].cat.codes

test_data['marital-status'] = test_data['marital-status'].astype('category')
test_data['marital-status'] = test_data['marital-status'].cat.codes

test_data['relationship'] = test_data['relationship'].astype('category')
test_data['relationship'] = test_data['relationship'].cat.codes

test_data['native-country'] = test_data['native-country'].astype('category')
test_data['native-country'] = test_data['native-country'].cat.codes
```

```
numeric_cols = ['age', 'workclass', 'education', 'education-num', 'marital-status',
       'occupation', 'relationship', 'race', 'sex', 'capital-gain',
       'capital-loss', 'native-country', 'hours-per-week']

for col in numeric_cols:
    col_data = test_data[col]
    index = numeric_cols.index(col)
    probs_morethan50 = [ norm.pdf(data, mean_people_with_more_than_50_salary[index], std_people_with_more_than_50_salary[inde
    probs_lessthan50 = [ norm.pdf(data, mean_people_with_less_than_50_salary[index], std_people_with_less_than_50_salary[inde

    diff = np.array(probs_morethan50)- np.array(probs_lessthan50)
    out = [("<=50K" if k<=0 else ">50K") for k in diff]

    s = 0
    td = test_data['salary']
    for i in range(col_data.size):
        if(td[i] == out[i]):
            s += 1
    print(col)
    print(s)
    print(s/col_data.size)
```

احتمال بالای 50 یا زیر 50 بودن دستمزد تحت هر شرایط یک ویژگی را بررسی و با هم مقایسه می کنیم، هر یک از احتمالات که بالاتر بود را به عنوان دستمزد آن فرد پیش بینی می کنیم و در out قرار می دهیم.

برای محاسبه خطا هم کافی است مقدار واقعی دستمزد را با مقدار پیش بینی شده مقایسه کنیم و تعداد پیش بینی ها غلط به کل را حساب کنیم.

تعداد موارد درست پیش بینی شده و درصد صحت پیش بینی هر یک از ویژگی ها را در خروجی زیر می بینیم:

```
age
1699
0.6209795321637427
------------------
workclass
1718
0.6279239766081871
------------------
education
1401
0.5120614035087719
------------------
education-num
1946
0.7112573099415205
------------------
marital-status
1902
0.6951754385964912
------------------
occupation
1489
0.5442251461988304
------------------
relationship
1422
0.5197368421052632
------------------
race
944
0.34502923976608185
------------------
```

```
------------------
sex
1369
0.5003654970760234
------------------
capital-gain
2139
0.7817982456140351
------------------
capital-loss
2071
0.7569444444444444
------------------
native-country
2049
0.7489035087719298
------------------
hours-per-week
1171
0.4279970760233918
------------------
```

همان طور که پیش بینی کرده بودیم، ویژگی capital-gain ویژگی دقیق تری است و با صحت 78 درصد و خطای حدودا 22 درصدی بهتر از بقیه می تواند salary را تعیین کند.

علل وجود خطا:

وجود خطا دلایل متعددی دارد مثلا بخشی از آن به این خاطر است که دیتای tarin کاملا ساده است و ممکن است به خوبی همه احتمالات را پوشش ندهد و با توجه به تعداد ستون ها تعداد داده کم باشد. و بخشی از خطا هم می تواند به خاطر این باشد که دستمزد تنها با یک ویژگی ارتباط ندارد و ترکیبی از ویژگی های متفاوت است و مثلا اگر دو ویژگی داشته باشیم که اثر معکوسی روی دستمزد داشته باشند و هر دو بالا باشند، تنها با تکیه بر یک ویژگی نمی توانیم دستمزد را به درستی پیش بینی کنیم.

در نهایت مقدار پیش بینی توسط ویژگی capital gain به همراه اندیس فرد متناظرش را به عنوان دستمزد فرد در یک فایل csv به نام salary prediction ذخیره می کنیم.

```
12 CAPITAL GAIN can predict salary better than other.
```

```
In [*]:  df = pd.DataFrame (best_predict, columns = ['salary'])
         df.to_csv('./salary_prediction.csv')
```

	A
1	,salary
2	0,<=50K
3	1,<=50K
4	2,<=50K
5	3,<=50K
6	4,<=50K
7	5,>50K
8	6,<=50K
9	7,<=50K
10	8,<=50K
11	9,<=50K
12	10,<=50K
13	11,>50K
14	12,<=50K
15	13,<=50K
16	14,<=50K
17	15,<=50K
18	16,<=50K
19	17,<=50K
20	18,<=50K
21	19,<=50K
22	20,<=50K
23	21,<=50K
24	22,>50K
25	23,<=50K
26	24,<=50K

salary_prediction

برخی منابع :

https://sparkbyexamples.com/pandas/pandas-write-dataframe-to-csv-file/#:~:text=By%20using%20pandas.,index%20as%20the%20first%20column.

https://datatofish.com/list-to-dataframe/

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html

https://stackoverflow.com/questions/10712002/

https://towardsdatascience.com/boolean-masking-with-pandas-b21b7714d0b6

https://www.jetbrains.com/help/pycharm/running-jupyter-notebook-cells.html

https://stackoverflow.com/questions/29803093/check-which-columns-in-dataframe-are-categorical