In [3]:
```
import pandas as pd

import pickle
```

In [4]:
```
df2=pd.read_csv('Dataset1.csv')
```

In [5]:
```
df3=df2
```

In [6]:
```
df2.head()
```

Out[6]:

	Job titiles	AI Impact	Tasks	AI models	AI_Workload_Ratio	Domain
0	Communications Manager	98%	365	2546	0.143362	Communication & PR
1	Data Collector	95%	299	2148	0.139199	Data & IT
2	Data Entry	95%	325	2278	0.142669	Administrative & Clerical
3	Mail Clerk	95%	193	1366	0.141288	Leadership & Strategy
4	Compliance Officer	92%	194	1369	0.141709	Medical & Healthcare

In [7]:
```
a1=df2['Job titiles']
```

In [8]:
```
df2.drop('Job titiles',axis=1,inplace=True)
```

In [9]:
```
df2.head()
```

Out[9]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Domain
0	98%	365	2546	0.143362	Communication & PR
1	95%	299	2148	0.139199	Data & IT
2	95%	325	2278	0.142669	Administrative & Clerical
3	95%	193	1366	0.141288	Leadership & Strategy
4	92%	194	1369	0.141709	Medical & Healthcare

In [10]:
```
df3 = pd.concat([df2,a1],axis=1)
```

In [11]:
```
df3.head()
```

Out[11]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Domain	Job titiles
0	98%	365	2546	0.143362	Communication & PR	Communications Manager
1	95%	299	2148	0.139199	Data & IT	Data Collector
2	95%	325	2278	0.142669	Administrative & Clerical	Data Entry
3	95%	193	1366	0.141288	Leadership & Strategy	Mail Clerk
4	92%	194	1369	0.141709	Medical & Healthcare	Compliance Officer

In [12]:
```
job=df3['Job titiles']
```

In [13]:
```
job.head()
```

Out[13]:
```
0    Communications Manager
1             Data Collector
2                 Data Entry
3                 Mail Clerk
4          Compliance Officer
Name: Job titiles, dtype: object
```

In [14]:
```
type(job)
```

Out[14]:
```
pandas.core.series.Series
```

In [15]:
```
jobtitles=[]
for i in job:
    jobtitles.append(i)
```

In [16]:
```
print(jobtitles)
```

```
ive', 'Accounting Clerk', 'Administrative Associate', 'Administrative Coor
dinator', 'Corporate Receptionist', 'Desk Receptionist', 'Executive Secret
ary', 'Front Desk Clerk', 'Front Desk Receptionist', 'Front Office Recepti
onist', 'Mail Sorter', 'Order Clerk', 'Medical Coder', 'Purchasing Assista
nt', 'Supply Technician', 'Android Developer', 'Ios Developer', 'Chief Of
Staff', 'Intelligence Analyst', 'Photo Retoucher', 'Full Charge Bookkeepe
r', 'Grant Accountant', 'Epic Analyst', 'It Project Manager', 'Software Pr
oject Manager', 'Technology Project Manager', 'Cobol Developer', 'Quantita
tive Analyst', 'Quantitative Research Analyst', 'Report Developer', 'Sas D
eveloper', 'Statistical Analyst', 'Backend Developer', 'Data Architect',
'Database Designer', 'Database Engineer', 'Gis Developer', 'Informatica De
veloper', 'Ui Developer', 'Ux Developer', 'Stenographer', 'Allocation Anal
yst', 'Business Operations Manager', 'Ecommerce', 'Solar Sales', 'Develope
r', 'Javascript Developer', 'Perl Developer', 'Pega Developer', 'Sap Devel
oper', 'Clinical Informatics Specialist', 'Cloud Engineer', 'Data Scientis
t', 'Database Programmer', 'Information Assurance', 'Oracle Developer', 'S
ql Database Developer', 'Sql Developer', 'Sql Server Developer', 'Teradata
Developer', 'Net Developer', 'Sports Analyst', 'Chief Of Police', 'Emergen
cy Dispatcher', 'Investigative Analyst', 'Police Dispatcher', 'Polygraph E
xaminer', 'Public Safety Dispatcher', 'Regulatory Analyst', 'Security Cons
```

In [17]:
```
from sklearn.preprocessing import LabelEncoder
```

In [18]:
```
le = LabelEncoder()
```

In [19]:
```
jobcode={}
```

In [20]:
```
j=1
for i in jobtitles:
    jobcode[i]=j
    j+=1
```

In [22]:
```
df3['job_title_code']=df3['Job titiles'].map(jobcode)
```

In [23]:
```
df3.head()
```

Out[23]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Domain	Job titiles	job_title_code
0	98%	365	2546	0.143362	Communication & PR	Communications Manager	1
1	95%	299	2148	0.139199	Data & IT	Data Collector	2
2	95%	325	2278	0.142669	Administrative & Clerical	Data Entry	3
3	95%	193	1366	0.141288	Leadership & Strategy	Mail Clerk	4
4	92%	194	1369	0.141709	Medical & Healthcare	Compliance Officer	5

In [24]:
```
df3=df3.drop('Domain',axis=1)
df3.head()
```

Out[24]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Job titiles	job_title_code
0	98%	365	2546	0.143362	Communications Manager	1
1	95%	299	2148	0.139199	Data Collector	2
2	95%	325	2278	0.142669	Data Entry	3
3	95%	193	1366	0.141288	Mail Clerk	4
4	92%	194	1369	0.141709	Compliance Officer	5

In [25]:
```
dff=df3.drop('Job titiles',axis=1)
```

In [26]:
```
df3
```

Out[26]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Job titiles	job_title_code
0	98%	365	2546	0.143362	Communications Manager	1
1	95%	299	2148	0.139199	Data Collector	2
2	95%	325	2278	0.142669	Data Entry	3
3	95%	193	1366	0.141288	Mail Clerk	4
4	92%	194	1369	0.141709	Compliance Officer	5
...	...	...	...	...	...	...
4701	5%	686	2798	0.245175	Singer	4702
4702	5%	556	2206	0.252040	Airport	4703
4703	5%	1316	4695	0.280298	Director	4704
4704	5%	710	2594	0.273709	Nurse	4705
4705	5%	825	3256	0.253378	Technician	4706

4706 rows × 6 columns

In [27]:
```
df3.head()
```

Out[27]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Job titiles	job_title_code
0	98%	365	2546	0.143362	Communications Manager	1
1	95%	299	2148	0.139199	Data Collector	2
2	95%	325	2278	0.142669	Data Entry	3
3	95%	193	1366	0.141288	Mail Clerk	4
4	92%	194	1369	0.141709	Compliance Officer	5

In [28]:
```
#Applying the model on the dataset
```

In [29]:
```
df3.head()
```

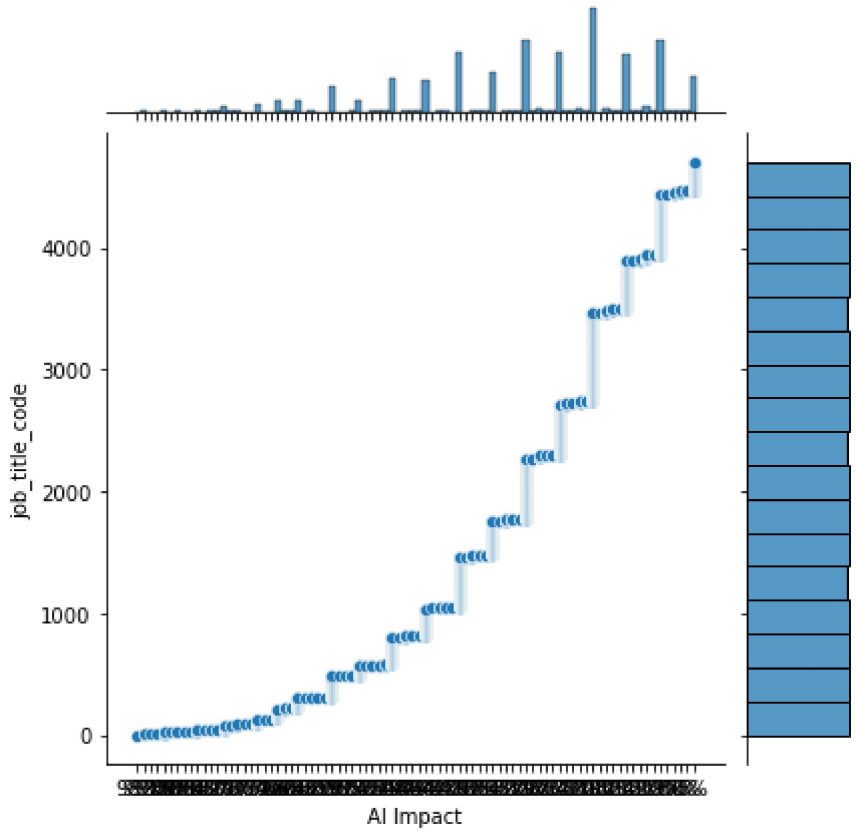
Out[29]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Job titiles	job_title_code
0	98%	365	2546	0.143362	Communications Manager	1
1	95%	299	2148	0.139199	Data Collector	2
2	95%	325	2278	0.142669	Data Entry	3
3	95%	193	1366	0.141288	Mail Clerk	4
4	92%	194	1369	0.141709	Compliance Officer	5

In [30]:
```
import seaborn as sns
```

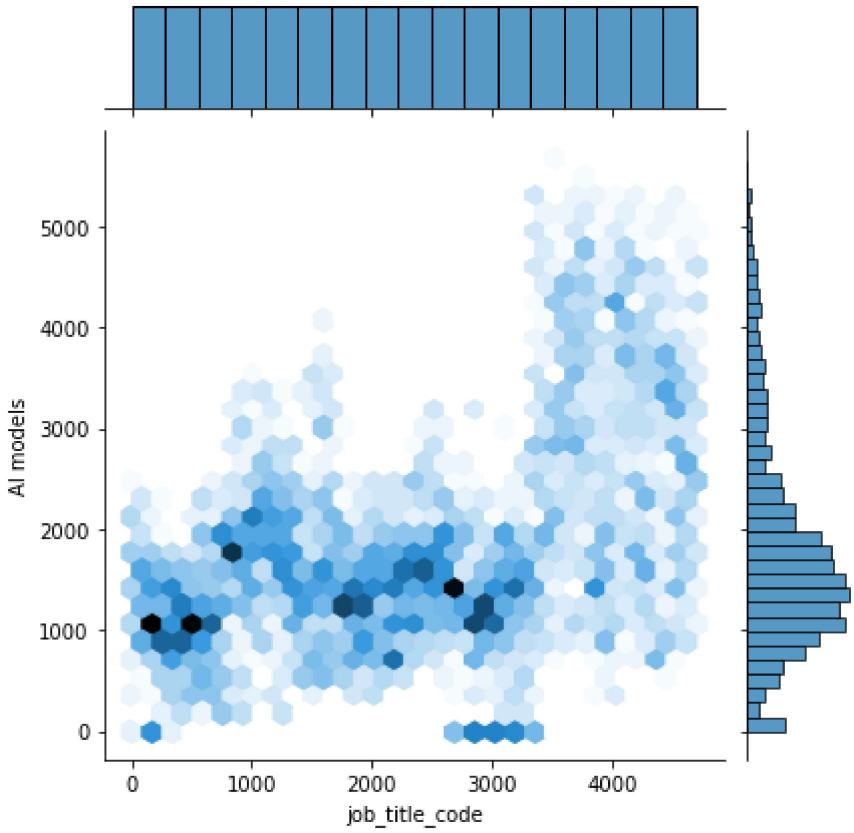
In [31]:
```
sns.jointplot(x='AI Impact',y='job_title_code',data=df3)
```

Out[31]:
```
<seaborn.axisgrid.JointGrid at 0x1e21de22d30>
```



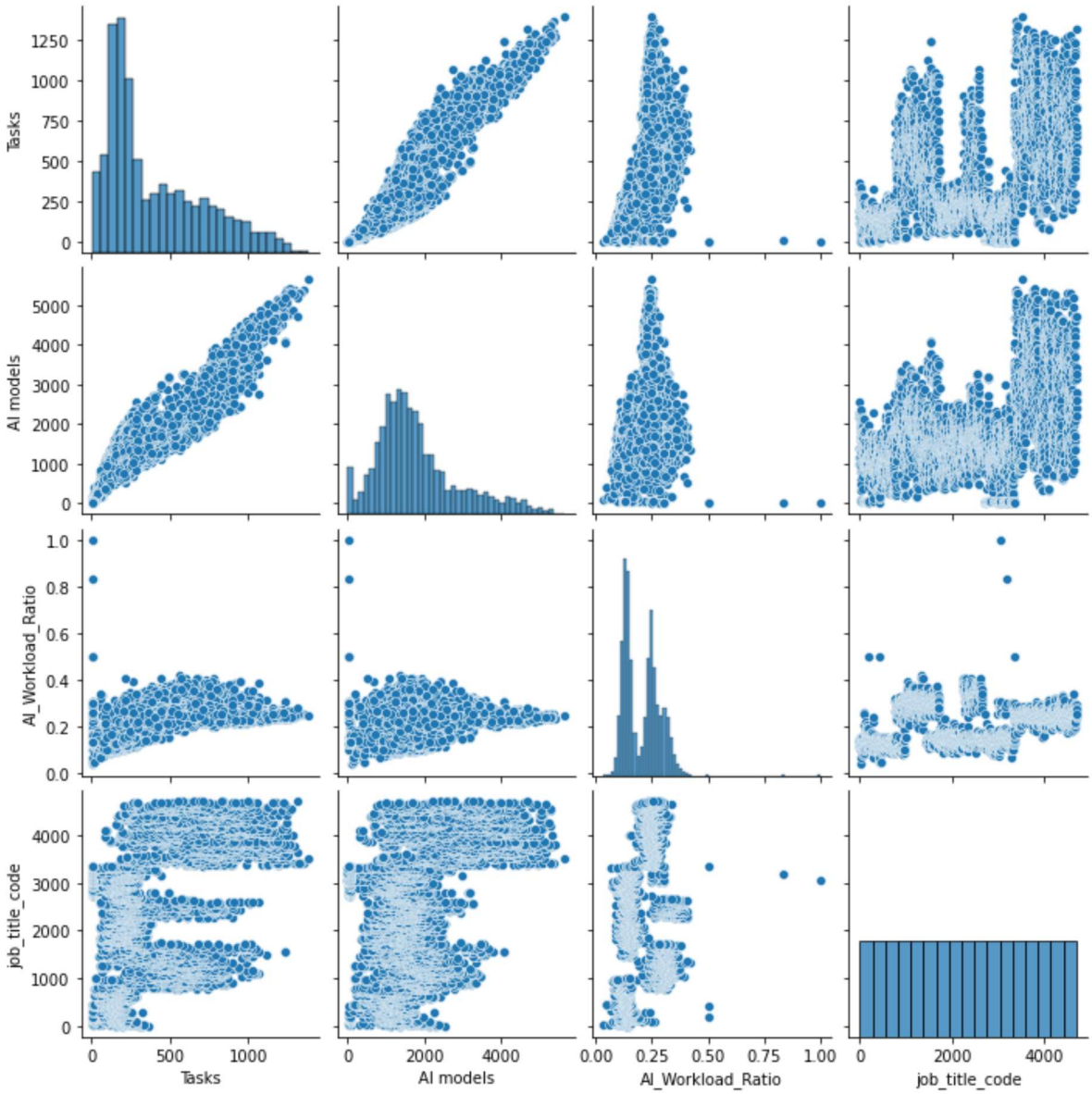
In [32]:
```
sns.jointplot(x='job_title_code',y='AI models',kind='hex',data=df3)
```

Out[32]:
```
<seaborn.axisgrid.JointGrid at 0x1e220f1d700>
```



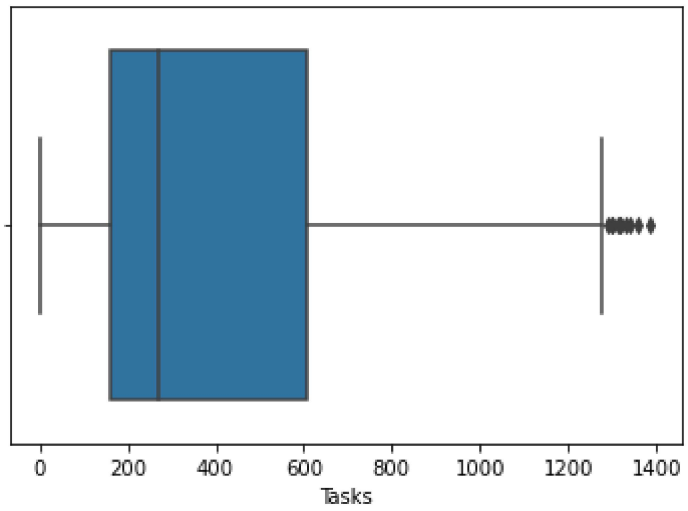
In [33]:
```
sns.pairplot(df3)
```

Out[33]:
```
<seaborn.axisgrid.PairGrid at 0x1e221194340>
```



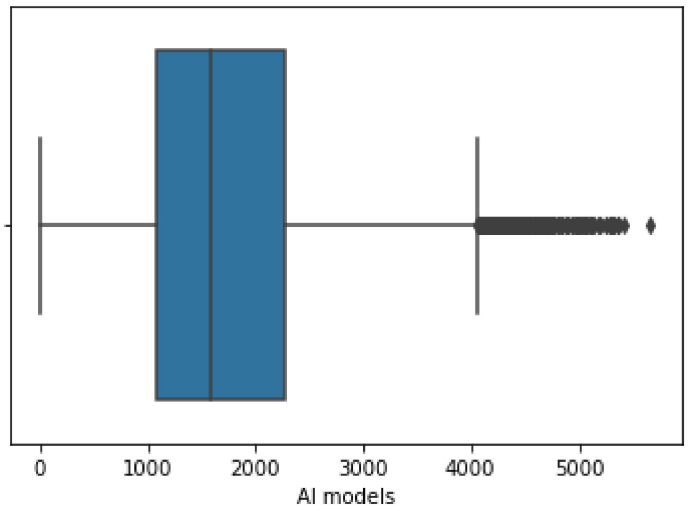
In [34]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='Tasks',data=df3)
```

Out[34]: <AxesSubplot:xlabel='Tasks'>



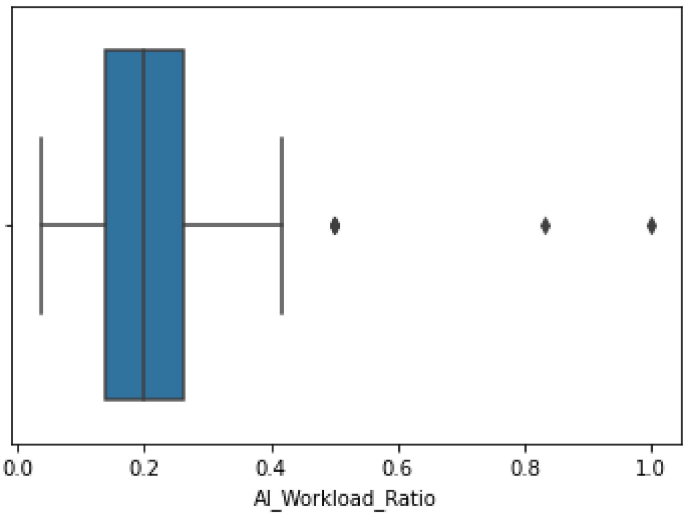
In [35]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='AI models',data=df3)
```

Out[35]: <AxesSubplot:xlabel='AI models'>



In [36]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='AI_Workload_Ratio',data=df3)
```

Out[36]: <AxesSubplot:xlabel='AI_Workload_Ratio'>

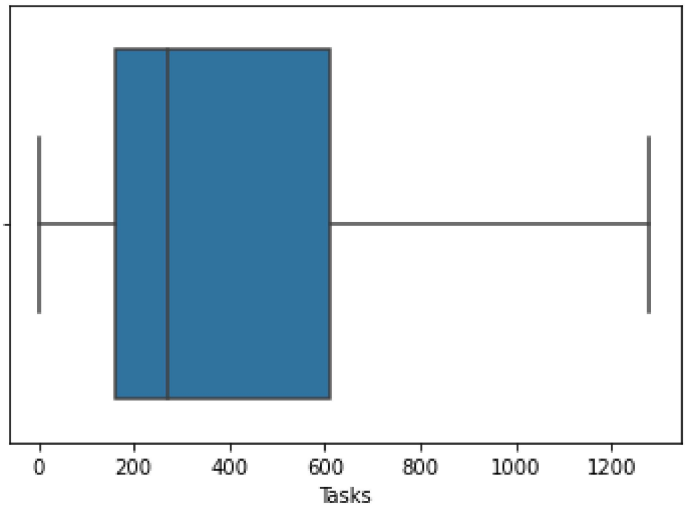


In [37]:
```
#REMOVING OUTLIERS
```

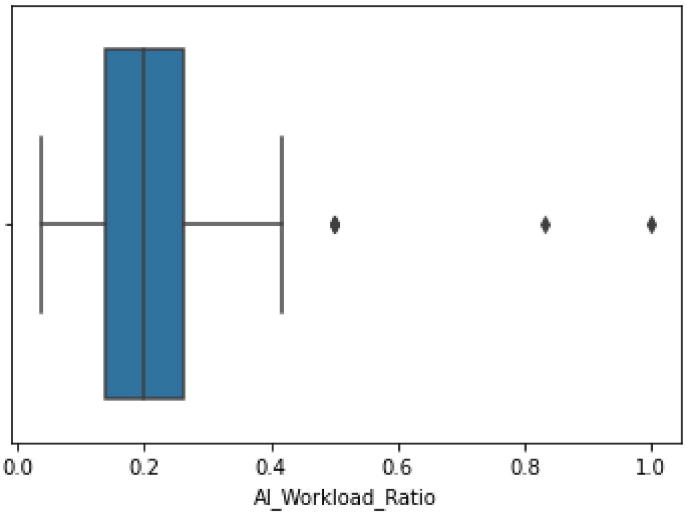
In [38]:
```
import numpy as np
```

In [39]:
```
percentile25=df3['Tasks'].quantile(0.25)
percentile75=df3['Tasks'].quantile(0.75)
iqr=percentile75-percentile25
upperlimitpm1=percentile75+1.5*iqr
lowerlimitpm1=percentile25-1.5*iqr
df3['Tasks']=np.where(df3['Tasks']>upperlimitpm1,upperlimitpm1,np.where(df3['T
```

In [40]:
```
sns.boxplot(x='Tasks',data=df3)
plt.show()
```



In [41]:
```
sns.boxplot(x='AI_Workload_Ratio',data=df3)
plt.show()
```



In [42]:
```
percentile25=df3['AI models'].quantile(0.25)
percentile75=df3['AI models'].quantile(0.75)
iqr=percentile75-percentile25
upperlimitpm1=percentile75+1.5*iqr
lowerlimitpm1=percentile25-1.5*iqr
df3['AI models']=np.where(df3['AI models']>upperlimitpm1,upperlimitpm1,np.wher
```

In [43]:
```
df3.head()
```

Out[43]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	Job titiles	job_title_code
0	98%	365.0	2546.0	0.143362	Communications Manager	1
1	95%	299.0	2148.0	0.139199	Data Collector	2
2	95%	325.0	2278.0	0.142669	Data Entry	3
3	95%	193.0	1366.0	0.141288	Mail Clerk	4
4	92%	194.0	1369.0	0.141709	Compliance Officer	5

In [44]:
```
df3=df3.drop('Job titiles',axis=1)
```

In [45]:
```
for i in range(len(df3['AI Impact'])):
    s=df3['AI Impact'][i]
    n=len(s)
    a=s[0:n-1]
    a=int(a)
    df3['AI Impact'][i]=a
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_36740\2622634572.py:6: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df3['AI Impact'][i]=a
```

In [46]:
```
df3.head()
```

Out[46]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	job_title_code
0	98	365.0	2546.0	0.143362	1
1	95	299.0	2148.0	0.139199	2
2	95	325.0	2278.0	0.142669	3
3	95	193.0	1366.0	0.141288	4
4	92	194.0	1369.0	0.141709	5

In [47]:
```
for i in range(len(df3['AI_Workload_Ratio'])):
    a=df3['AI_Workload_Ratio'][i]
    b=round(a,2)
    df3['AI_Workload_Ratio'][i]=b
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_36740\3517472379.py:4: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df3['AI_Workload_Ratio'][i]=b
```

In [48]:
```
import numpy as np

# Check for NaN values
nan_indices = np.isnan(df3['AI_Workload_Ratio'])

# Check for infinite values
inf_indices = np.isinf(df3['AI_Workload_Ratio'])

# Print the indices where NaN or infinite values are present
print("NaN indices:", np.where(nan_indices))
print("Infinite indices:", np.where(inf_indices))
```

```
NaN indices: (array([], dtype=int64),)
Infinite indices: (array([3034, 3035, 3036, 3037, 3184, 3211, 3322], dtype=in
t64),)
```

In [49]:
```
df3['AI_Workload_Ratio'][3034]
```

Out[49]:
```
inf
```

In [50]:
```
df3.replace([np.inf, -np.inf], np.nan, inplace=True)
```

In [51]:
```
import numpy as np

# Check for NaN values
nan_indices = np.isnan(df3['Tasks'])

# Check for infinite values
inf_indices = np.isinf(df3['Tasks'])

# Print the indices where NaN or infinite values are present
print("NaN indices:", np.where(nan_indices))
print("Infinite indices:", np.where(inf_indices))
```

```
NaN indices: (array([], dtype=int64),)
Infinite indices: (array([], dtype=int64),)
```

In [52]:
```
df3=df3.dropna()
```

In [53]:
```
df3.head()
```

Out[53]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	job_title_code
0	98	365.0	2546.0	0.14	1
1	95	299.0	2148.0	0.14	2
2	95	325.0	2278.0	0.14	3
3	95	193.0	1366.0	0.14	4
4	92	194.0	1369.0	0.14	5

In [ ]:

In [ ]:

In [89]:
```
from sklearn.model_selection import train_test_split
```

In [90]:
```
X_train, X_test, y_train, y_test = train_test_split(df3.drop('AI Impact',axis=
                                                    df3['AI Impact'], test_siz
```

In [91]:
```
#from sklearn.preprocessing import MinMaxScaler

# Assuming 'data' is your DataFrame or array
# Replace 'data' with your actual data

# Create a MinMaxScaler
#scaler = MinMaxScaler()

# Fit the scaler on the data and transform the data
#X_train_normalized = scaler.fit_transform(X_train)
#X_test_normalized = scaler.transform(X_test)
```

In [ ]:

In [149]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [177]:
```
knn = KNeighborsClassifier(n_neighbors=12)
```

In [178]:
```
knn.fit(X_train.values,y_train.values)
```

Out[178]:
```
KNeighborsClassifier(n_neighbors=12)
```

In [179]:
```
y_pred = knn.predict(X_test.values)
```

In [180]:
```
print(y_pred)
```

```
[40 40 50 ... 15 10  5]
```

In [ ]:

In [181]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [182]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, y_pred)
print(mae)
```

```
0.4929078014184397
```

In [183]:
```
from sklearn.metrics import r2_score

r2=r2_score(y_test,y_pred)
print(r2)
```

```
0.9901979676230852
```

In [184]:
```
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
print(mape)
```

```
1.4301356914574526
```

In [185]:
```
from sklearn.metrics import explained_variance_score
explained_var = explained_variance_score(y_test, y_pred)
print(explained_var)
```

```
0.9903415336926783
```

In [186]:
```
filename = "jobpredict.pkl"
```

In [187]:
```
pickle.dump(knn,open(filename,'wb'))
```

In [188]:
```
knn1=pickle.load(open('jobpredict.pkl','rb'))
```

In [189]:
```
print(knn1.predict([[323.0,1354.0,0.14,1]]))
```

```
[85]
```

In [ ]:

In [ ]: