In [3]:

```
import pandas as pd
```

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)
```

```
Analyst', 'It Specialist', 'Junior Network Engineer', 'Network Consultant', 'Network Support Specialist', 'Salesforce Adminis
trator', 'Sap Functional Consultant', 'Support Technician', 'Crisis Counselor', 'Mechanical Drafter', 'Production Engineer',
'Security Engineer', 'Simulation Engineer', 'Underground Miner', 'Unix Engineer', 'Verification Engineer', 'Voice Engineer',
'Chief Administrative Officer', 'Emergency Management Specialist', 'Loss Prevention Manager', 'Proposal Coordinator', 'Provid
er Relations Representative', 'Secretary', 'Trading Assistant', 'Volunteer Coordinator', 'Car Driver', 'Carrier', 'Constructi
on Driver', 'Contract Driver', 'Courier', 'Forklift Driver', 'Tanker', 'Taxi Driver', 'Assistant Restaurant Manager', 'Banque
t Captain', 'Brewer', 'Chef Manager', 'Executive Chef', 'Food Production Worker', 'Front Of House Manager', 'Front Office Sup
ervisor', 'Greeter', 'Hotel Front Office Manager', 'Hotel General Manager', 'Hotel Operations Manager', 'Contract Negotiato
r', 'Housing Specialist', 'Proposal Specialist', 'Analytical Chemist', 'Cytogenetic Technologist', 'Food Scientist', 'Food Te
chnologist', 'Laboratory Assistant', 'Patent Agent', 'Taxonomist', 'Accounting Consultant', 'Accounting Technician', 'Account
s Receivable Clerk', 'Appraiser', 'Broker Assistant', 'Commercial Loan Officer', 'Commodity Trader', 'Corporate Accountant',
'Entry Level Accountant', 'Equity Trader', 'Financial Accountant', 'Fund Accounting Manager', 'General Accountant', 'General
Ledger Accountant', 'Investment Banking Analyst', 'Private Equity Analyst', 'Risk Analyst', 'Treasury Manager', 'Clinic Recep
tionist', 'Dental Receptionist', 'Front Office Assistant', 'Yard Jockey', 'Associate Consultant', 'Consultant', 'Financial Co
nsultant', 'Technical Consultant', 'Technology Consultant', 'Product Demonstrator', 'Promotions Manager', 'Purchasing Agent',
'Special Events Coordinator', 'Assistant Project Manager', 'Associate Product Manager', 'Product Director', 'Product Owner',
'Project Estimator', 'Project Lead', 'Project Leader', 'Project Management Specialist', 'Inventory Specialist', 'Merchandisin
g Assistant', 'Stock Manager', 'Architectural Drafter', 'Drilling Supervisor', 'General Contractor', 'Safety Professional',
'Safety Specialist', 'Adjuster', 'Claim Specialist', 'Claims Examiner', 'Claims Representative', 'Insurance Broker', 'Policy
Advisor', 'Mobile Phlebotomist', 'Orthopedic Technician', 'Aircraft Assembler', 'Aircraft Maintenance Technician', 'Aircraft
```

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 [21]:
```
print(jobcode)
```

```
ce Analyst': 331, 'Risk Management Analyst': 332, 'Salesforce Business Analyst': 333, 'Data Analyst': 334, 'Database Architec
t': 335, 'Health Data Analyst': 336, 'Sql Dba': 337, 'Sql Server Dba': 338, 'Shipping Clerk': 339, 'Timekeeper': 340, 'Admini
stration': 341, 'Administrative Director': 342, 'Assistant Administrator': 343, 'Compliance Analyst': 344, 'Front Desk Coordi
nator': 345, 'Office Clerk': 346, 'Operations Clerk': 347, 'Procurement Clerk': 348, 'Receiver': 349, 'Registration Clerk': 3
50, 'Registration Specialist': 351, 'Unit Secretary': 352, 'Healthcare Business Analyst': 353, 'Surgery Scheduler': 354, 'Pos
t Office': 355, 'Demand Planner': 356, 'Logistics Coordinator': 357, 'Procurement Agent': 358, 'Purchasing Coordinator': 359,
'Shipping Coordinator': 360, 'Supply Chain Coordinator': 361, 'Supply Coordinator': 362, 'Assignment Editor': 363, 'Front Des
k Agent': 364, 'Radio Operator': 365, 'Manual Qa Tester': 366, 'Manual Tester': 367, 'Coldfusion Developer': 368, 'Computer S
cientist': 369, 'Drupal Developer': 370, 'Java': 371, 'Java Developer': 372, 'Java Engineer': 373, 'Java Programmer': 374, 'J
ava Software Developer': 375, 'Home Inspector': 376, 'Biomedical Engineer': 377, 'Biotechnology': 378, 'Environmental Enginee
r': 379, 'Industrial Organizational Psychologist': 380, 'Accounts Payable': 381, 'Auditor': 382, 'Budget Analyst': 383, 'Coll
ection Agent': 384, 'Collection Representative': 385, 'Collection Specialist': 386, 'Collections Specialist': 387, 'Complianc
e Auditor': 388, 'Cost Controller': 389, 'Credit Controller': 390, 'Debt Collector': 391, 'Exchange Engineer': 392, 'Financia
l Engineer': 393, 'Financial Examiner': 394, 'Loan Closer': 395, 'Mortgage Closer': 396, 'Mortgage Loan Closer': 397, 'Proces
sor': 398, 'Trader': 399, 'Treasury Accountant': 400, 'Clinical Analyst': 401, 'Meter Reader': 402, 'Gas Station Attendant':
403, 'Lactation Consultant': 404, 'Ticket Taker': 405, 'Usher': 406, '911 Dispatcher': 407, '911 Operator': 408, 'Fingerprint
Technician': 409, 'Forensic Examiner': 410, 'Forensic Scientist': 411, 'Intelligence': 412, 'Military Analyst': 413, 'Safety
Coordinator': 414, 'Security Technician': 415, 'Skip Tracer': 416, 'Information Analyst': 417, 'Information Specialist': 418,
'Network Systems Administrator': 419, 'Senior Systems Engineer': 420, 'Software Engineer': 421, 'Systems Administrator': 422,
'Systems Analyst': 423, 'Unix Administrator': 424, 'Unix System Administrator': 425, 'Vmware Administrator': 426, 'Windows Ad
```

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()
```

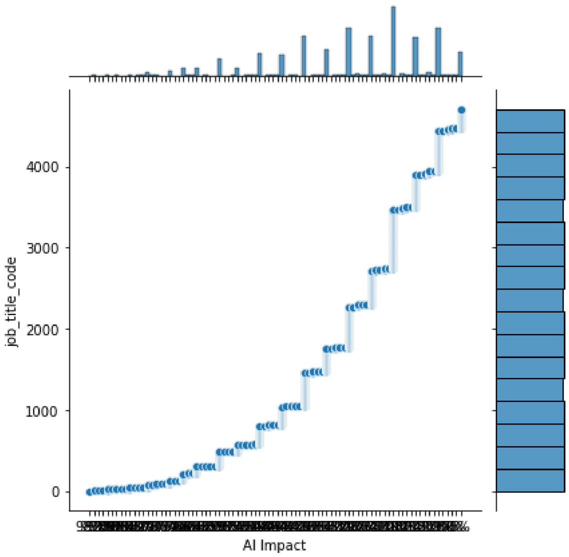
Out[29]:

	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 [31]:
```

import seaborn as sns
```

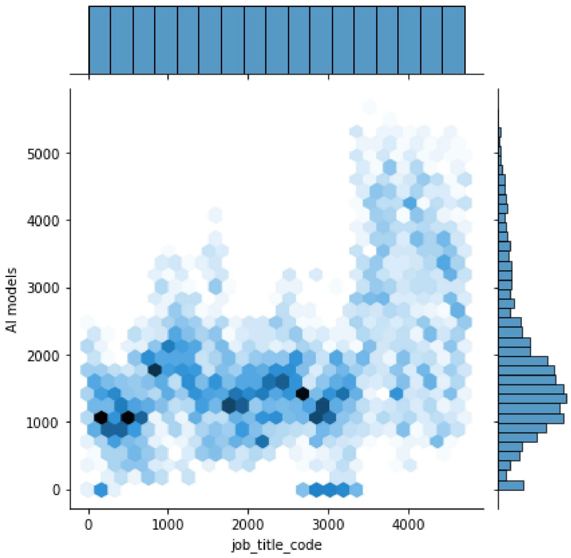
```
In [32]: sns.jointplot(x='AI Impact',y='job_title_code',data=df3)
```

```
Out[32]: <seaborn.axisgrid.JointGrid at 0x258f5faffa0>
```



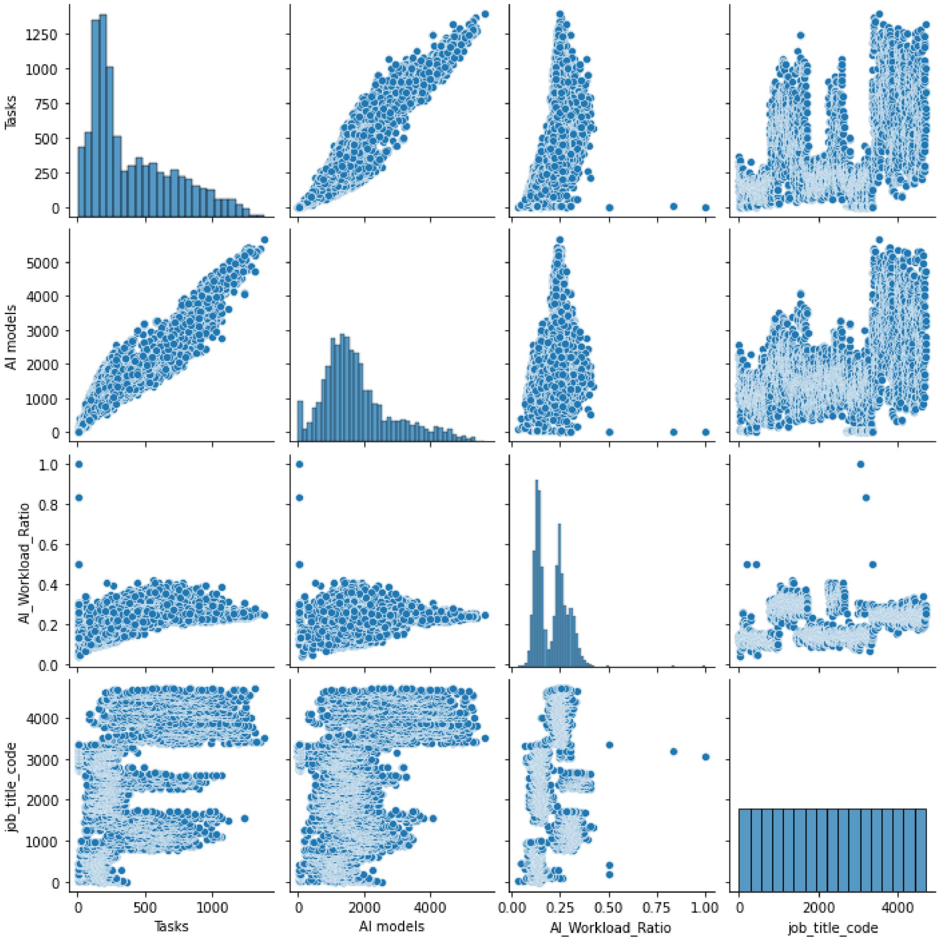
```
In [33]: sns.jointplot(x='job_title_code',y='AI models',kind='hex',data=df3)
```

```
Out[33]: <seaborn.axisgrid.JointGrid at 0x25892f38580>
```



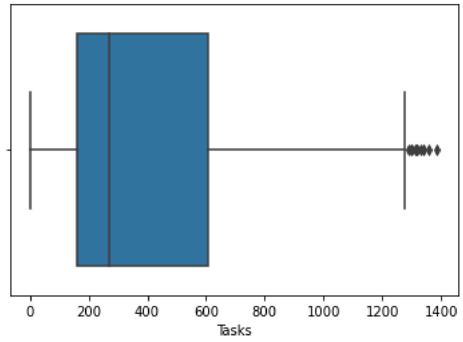
```
In [34]: sns.pairplot(df3)
```

```
Out[34]: <seaborn.axisgrid.PairGrid at 0x258943d2760>
```

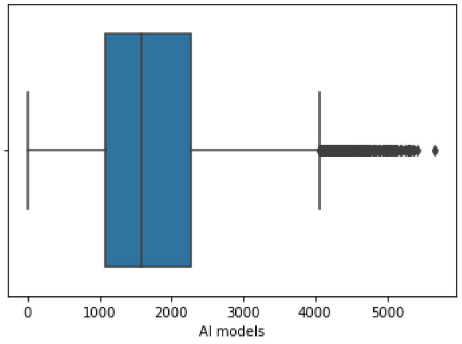


```
In [35]: import seaborn as sns
         import matplotlib.pyplot as plt
         sns.boxplot(x='Tasks',data=df3)
```

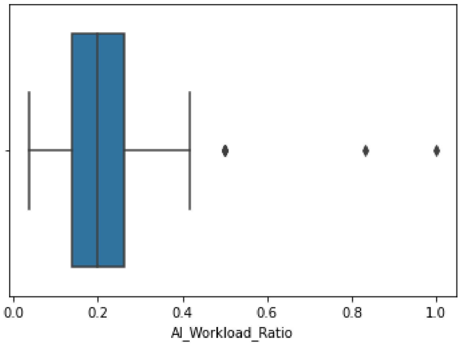
```
Out[35]: <AxesSubplot:xlabel='Tasks'>
```



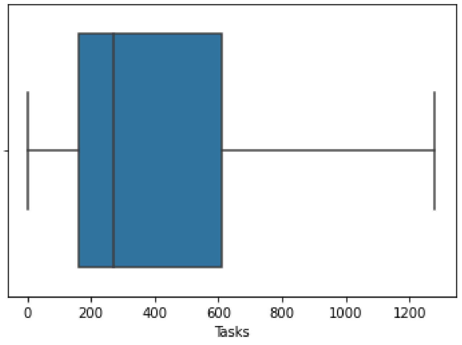
```
In [36]: import seaborn as sns
         import matplotlib.pyplot as plt
         sns.boxplot(x='AI models',data=df3)

Out[36]: <AxesSubplot:xlabel='AI models'>
```



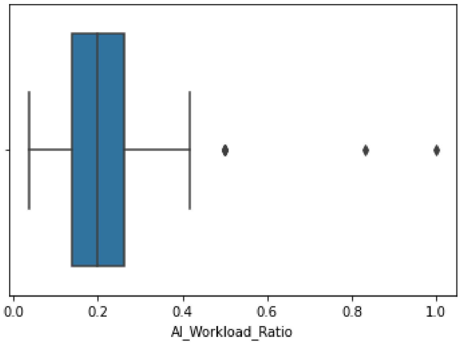
```
In [38]: import seaborn as sns
         import matplotlib.pyplot as plt
         sns.boxplot(x='AI_Workload_Ratio',data=df3)

Out[38]: <AxesSubplot:xlabel='AI_Workload_Ratio'>
```



```
In [40]: #REMOVING OUTLIERS

In [42]: import numpy as np

In [43]: 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['Tasks']<lowerlimitpm1,lowerlimitpm1,df3['Tasks']))

In [44]: sns.boxplot(x='Tasks',data=df3)
         plt.show()
```



In [60]:
```
sns.boxplot(x='AI_Workload_Ratio',data=df3)
plt.show()
```



In [45]:
```
5=df3['AI models'].quantile(0.25)
5=df3['AI models'].quantile(0.75)
ile75-percentile25
m1=percentile75+1.5*iqr
m1=percentile25-1.5*iqr
els']=np.where(df3['AI models']>upperlimitpm1,upperlimitpm1,np.where(df3['AI models']<lowerlimitpm1,lowerlimitpm1,df3['AI models'
```

In [47]:
```
df3.head()
```

Out[47]:

	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 [49]:
```
df3=df3.drop('Job titiles',axis=1)
```

In [54]:
```
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_23032\2622634572.py:6: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df3['AI Impact'][i]=a
```

In [55]:
```
df3.head()
```

Out[55]:

	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 [63]:
```
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_23032\3517472379.py:4: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df3['AI_Workload_Ratio'][i]=b
```

In [67]:

```
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=int64),)
```

In [74]:

```
df3['AI_Workload_Ratio'][3034]
```

Out[74]:

```
inf
```

In [75]:

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

In [76]:

```
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 [78]:

```
df3=df3.dropna()
```

In [79]:

```
df3.head()
```

Out[79]:

	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 [80]:

In [ ]:

In [81]:

```
from sklearn.model_selection import train_test_split
```

In [82]:

```
X_train, X_test, y_train, y_test = train_test_split(df3.drop('job_title_code',axis=1),
                                                    df3['job_title_code'], test_size=0.30)
```

In [105]:

```
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 [106]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [107]: knn = KNeighborsClassifier(n_neighbors=1)
```

```
In [108]: knn.fit(X_train_normalized,y_train)

Out[108]: KNeighborsClassifier(n_neighbors=1)
```

```
In [109]: y_pred = knn.predict(X_test_normalized)
```

```
In [111]: from sklearn.metrics import classification_report,confusion_matrix
```

```
In [116]: from sklearn.metrics import mean_absolute_error
          mae = mean_absolute_error(y_test, y_pred)
          print(mae)

          95.92056737588652
```

```
In [113]: from sklearn.metrics import r2_score

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

          0.9872609984254223
```

```
In [120]: mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
          print(mape)

          6.025627249843691
```

```
In [121]: from sklearn.metrics import explained_variance_score
          explained_var = explained_variance_score(y_test, y_pred)
          print(explained_var)

          0.9872624088748995
```

```
In [ ]:
```