In [220]:
```
import pandas as pd
```

In [221]:
```
df  = pd.read_csv('Dataset1.csv')
```

In [222]:
```
df.head()
```

Out[222]:

	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 [223]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```

```

In [224]:
```
domain = df['Domain']
```

In [225]:
```
len(domain)
```

Out[225]:
```
4706
```

In [226]:
```
new_domain  = set(domain)
```

In [227]:
```
len(new_domain)
```

Out[227]:
```
10
```

In [228]:
```
print(new_domain)
```

```
{'Supply Chain & Logistics', 'Administrative & Clerical', 'Law Enforcement',
'Hospitality', 'Data & IT', 'Medical & Healthcare', 'Communication & PR', 'Sa
les & Marketing', 'Construction', 'Leadership & Strategy'}
```

In [229]:
```
domains=[]
for i in new_domain:
    domains.append(i)
```

In [230]:
```
df.head()
```

Out[230]:

	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 [ ]:
```
```

In [231]:
```
df.head()
```

Out[231]:

	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 [232]:
```
job=df['Job titiles']
jobtitles=[]
for i in job:
    jobtitles.append(i)
jobcode={}
j=1
for i in jobtitles:
    jobcode[i]=j
    j+=1

df['job_title_code']=df['Job titiles'].map(jobcode)
```

In [ ]:
```
```

In [ ]:
```
```

In [ ]:

```
```

In [233]:

```
domaincode={}
```

In [234]:

```
j=1
for i in domains:
    domaincode[i]=j
    j+=1
```

In [ ]:

```
```

In [235]:

```
df['Domain_Code']=df['Domain'].map(domaincode)
```

In [236]:

```
df.head()
```

Out[236]:

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

In [237]:

```
df = df.drop('Domain',axis=1)
```

In [238]:

```
df.head()
```

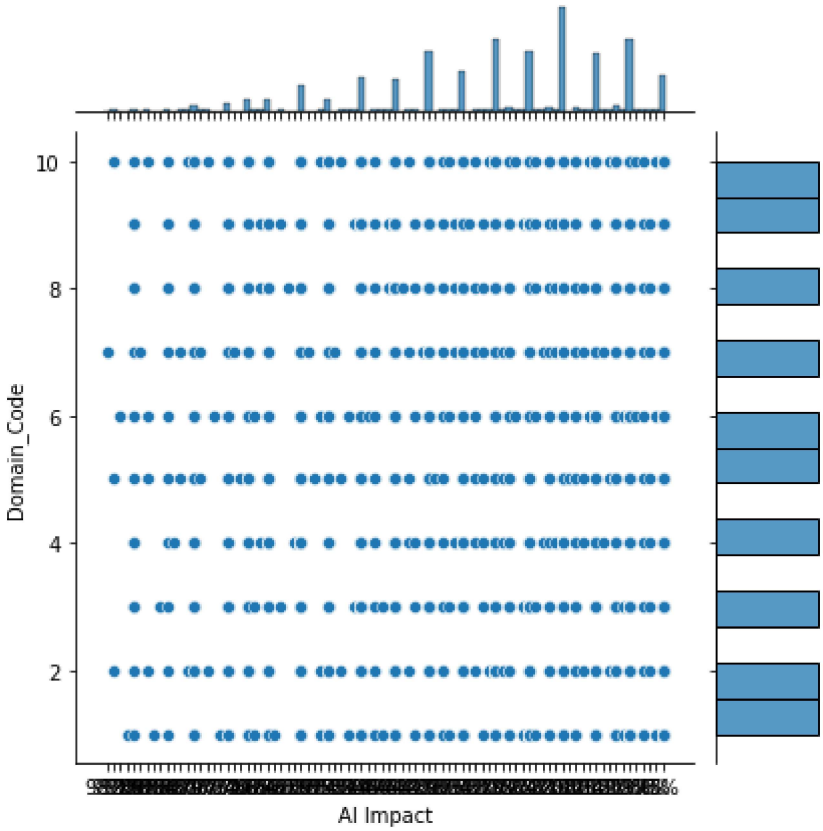
Out[238]:

	Job titiles	AI Impact	Tasks	AI models	AI_Workload_Ratio	job_title_code	Domain_Code
0	Communications Manager	98%	365	2546	0.143362	1	7
1	Data Collector	95%	299	2148	0.139199	2	5
2	Data Entry	95%	325	2278	0.142669	3	2
3	Mail Clerk	95%	193	1366	0.141288	4	10
4	Compliance Officer	92%	194	1369	0.141709	5	6

In [239]:
```
import seaborn as sns
```

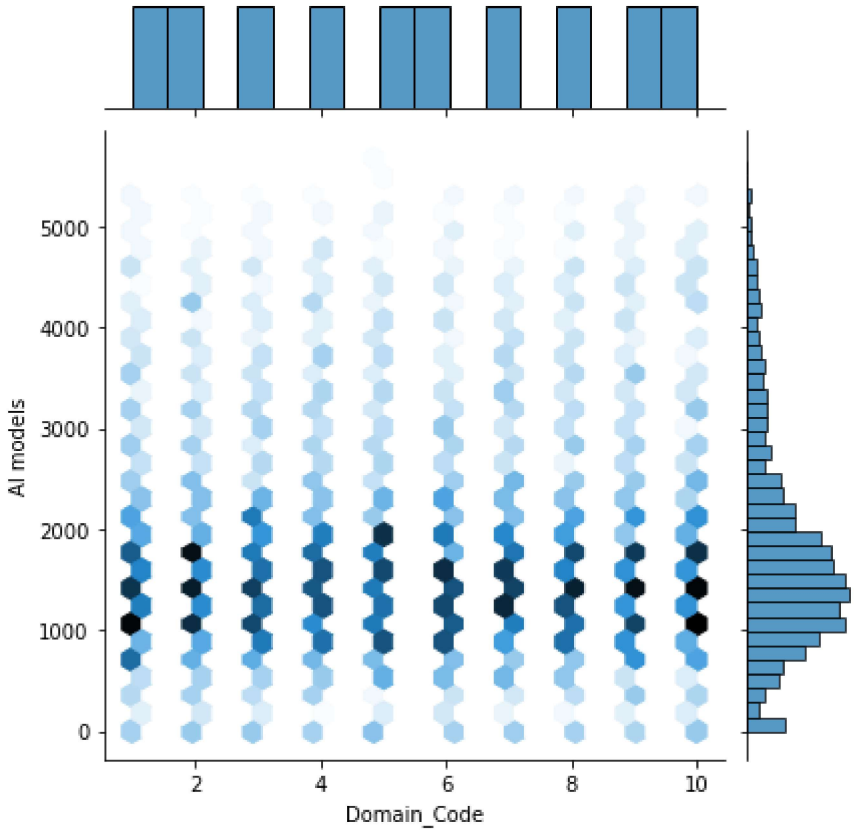
In [240]:
```
sns.jointplot(x='AI Impact',y='Domain_Code',data=df)
```

Out[240]:
```
<seaborn.axisgrid.JointGrid at 0x1ea0a5b2940>
```



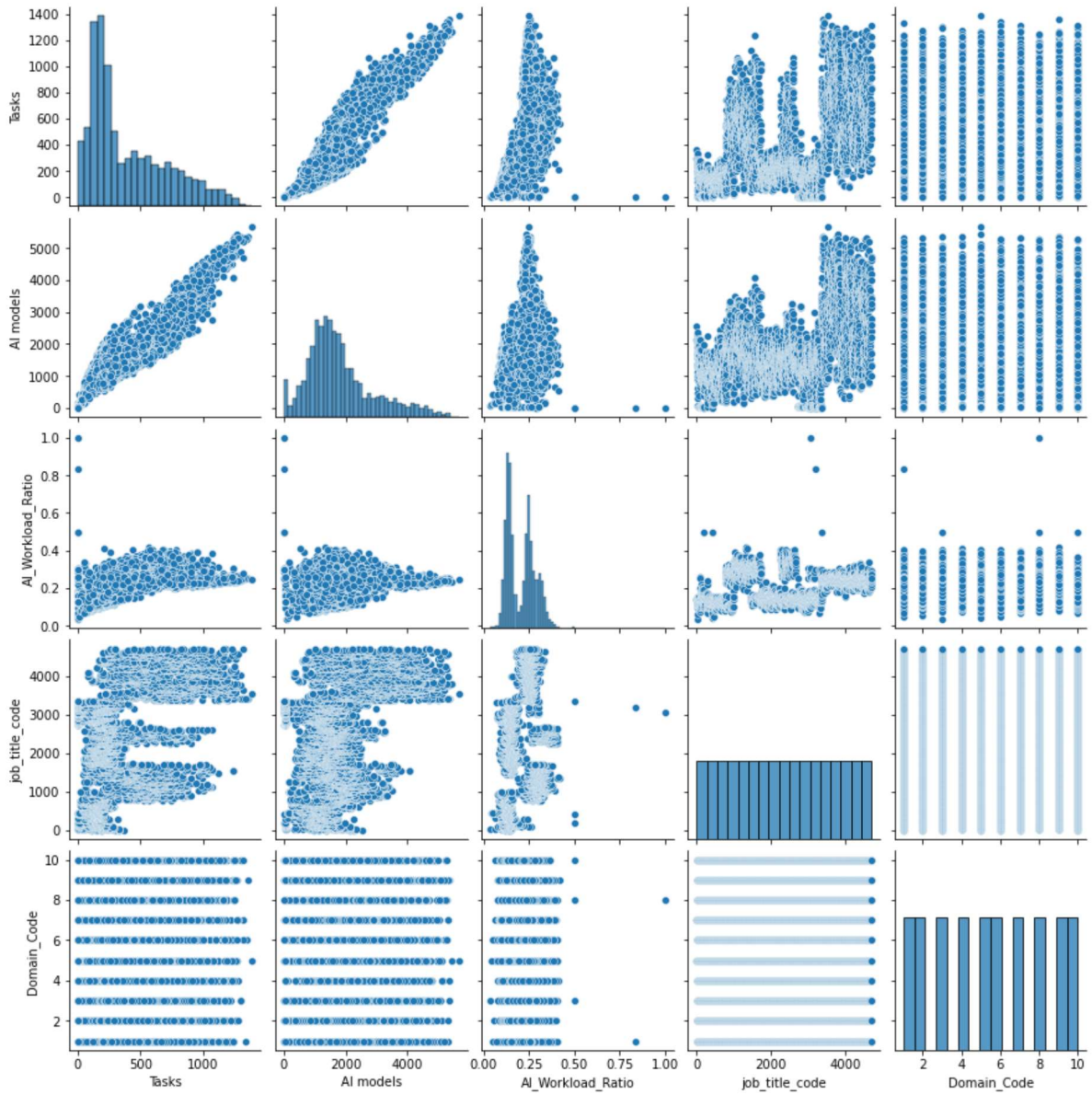
In [241]:
```
sns.jointplot(x='Domain_Code',y='AI models',kind='hex',data=df)
```

Out[241]:
```
<seaborn.axisgrid.JointGrid at 0x1ea0d0602e0>
```



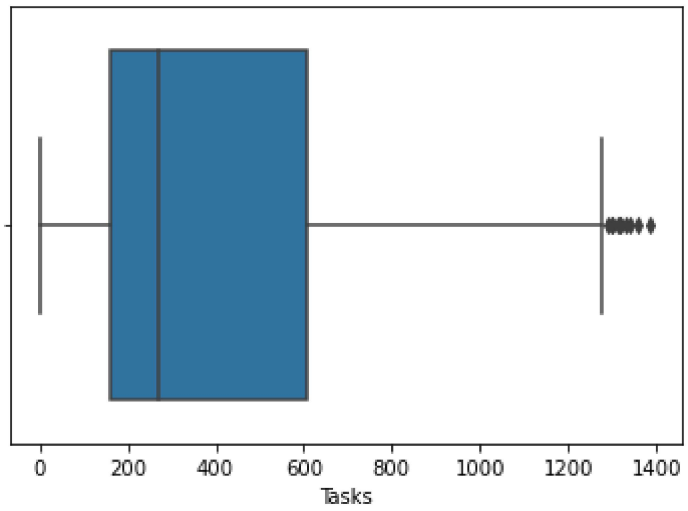
In [242]:
```
sns.pairplot(df)
```

Out[242]: <seaborn.axisgrid.PairGrid at 0x1ea0d1c6310>



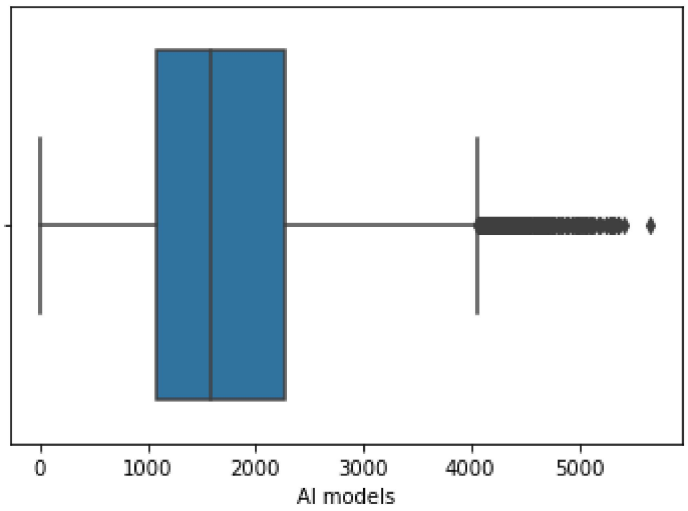
In [243]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='Tasks',data=df)
```

Out[243]: <AxesSubplot:xlabel='Tasks'>



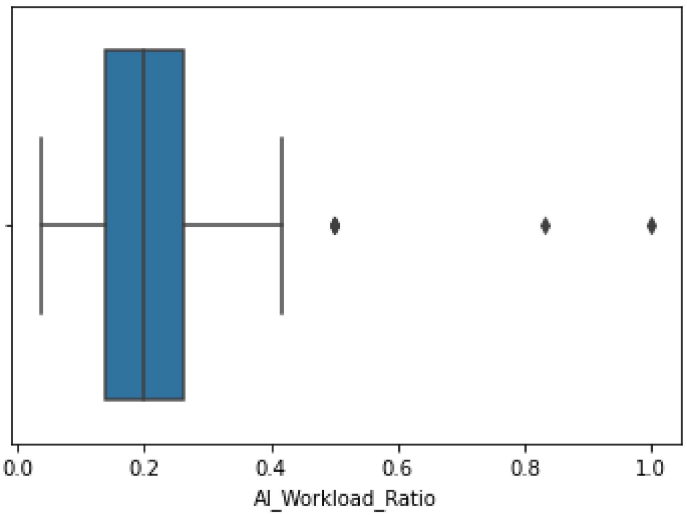
In [244]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='AI models',data=df)
```

Out[244]: <AxesSubplot:xlabel='AI models'>



In [245]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='AI_Workload_Ratio',data=df)
```

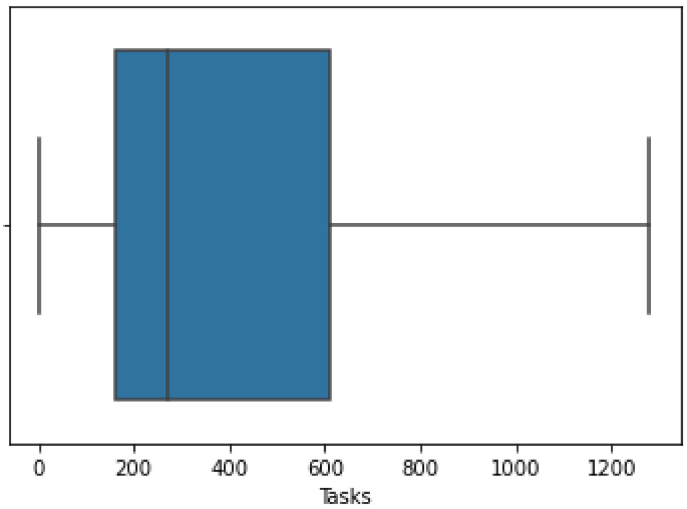
Out[245]: <AxesSubplot:xlabel='AI_Workload_Ratio'>



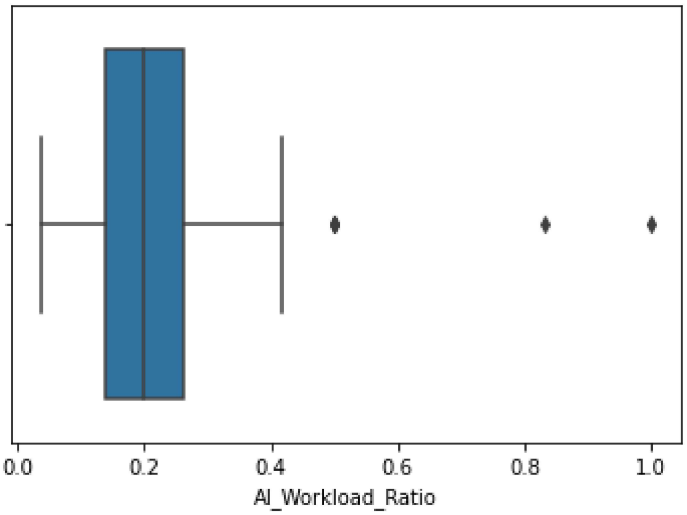
In [246]:
```
import numpy as np
```

In [247]:
```
percentile25=df['Tasks'].quantile(0.25)
percentile75=df['Tasks'].quantile(0.75)
iqr=percentile75-percentile25
upperlimitpm1=percentile75+1.5*iqr
lowerlimitpm1=percentile25-1.5*iqr
df['Tasks']=np.where(df['Tasks']>upperlimitpm1,upperlimitpm1,np.where(df['Task
```

In [248]:
```
sns.boxplot(x='Tasks',data=df)
plt.show()
```



In [249]:
```
sns.boxplot(x='AI_Workload_Ratio',data=df)
plt.show()
```



In [250]:
```
percentile25=df['AI models'].quantile(0.25)
percentile75=df['AI models'].quantile(0.75)
iqr=percentile75-percentile25
upperlimitpm1=percentile75+1.5*iqr
lowerlimitpm1=percentile25-1.5*iqr
df['AI models']=np.where(df['AI models']>upperlimitpm1,upperlimitpm1,np.where(
```

In [251]:
```
df.head()
```

Out[251]:

	Job titiles	AI Impact	Tasks	AI models	AI_Workload_Ratio	job_title_code	Domain_Code
0	Communications Manager	98%	365.0	2546.0	0.143362	1	7
1	Data Collector	95%	299.0	2148.0	0.139199	2	5
2	Data Entry	95%	325.0	2278.0	0.142669	3	2
3	Mail Clerk	95%	193.0	1366.0	0.141288	4	10
4	Compliance Officer	92%	194.0	1369.0	0.141709	5	6

In [252]:
```
df=df.drop('Job titiles',axis=1)
```

In [253]:
```
for i in range(len(df['AI Impact'])):
    s=df['AI Impact'][i]
    n=len(s)
    a=s[0:n-1]
    a=int(a)
    df['AI Impact'][i]=a
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_18756\2618703470.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)
  df['AI Impact'][i]=a
```

In [254]:
```
for i in range(len(df['AI_Workload_Ratio'])):
    a=df['AI_Workload_Ratio'][i]
    b=round(a,2)
    df['AI_Workload_Ratio'][i]=b
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_18756\4022844787.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)
  df['AI_Workload_Ratio'][i]=b
```

In [255]:
```
import numpy as np

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

# Check for infinite values
inf_indices = np.isinf(df['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 [256]:
```
df.replace([np.inf, -np.inf], np.nan, inplace=True)
```

In [257]:
```
import numpy as np

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

# Check for infinite values
inf_indices = np.isinf(df['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 [258]:
```
df=df.dropna()
```

In [259]:
```
df.head()
```

Out[259]:

	AI Impact	Tasks	AI models	AI_Workload_Ratio	job_title_code	Domain_Code
0	98	365.0	2546.0	0.14	1	7
1	95	299.0	2148.0	0.14	2	5
2	95	325.0	2278.0	0.14	3	2
3	95	193.0	1366.0	0.14	4	10
4	92	194.0	1369.0	0.14	5	6

In [ ]:
```

```

In [ ]:
```

```

In [260]:
```
from sklearn.model_selection import train_test_split
```

In [261]:
```
X_train, X_test, y_train, y_test = train_test_split(df.drop('AI Impact',axis=1
                                                    df['AI Impact'], test_size
```

In [ ]:
```

```

In [262]:
```
##KNN Regressor
```

In [263]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [264]:
```
knn = KNeighborsRegressor(n_neighbors=120)
```

In [265]:
```
knn.fit(X_train.values,y_train.values)
```

Out[265]:
```
KNeighborsRegressor(n_neighbors=120)
```

In [266]:
```
y_pred = knn.predict(X_test.values)
```

In [267]:
```
print(y_pred)
```

```
[35.41666667 38.975      15.69166667 ... 14.93333333 40.39166667
 54.475     ]
```

In [268]:
```
from sklearn.metrics import r2_score

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

```
0.9717996351883951
```

In [269]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, y_pred)
print(mae)
```

```
1.7300059101654848
```

In [270]:
```
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
print(mape)
```

```
7.843821180328141
```

In [ ]:
```

```

In [271]:
```
##SVR MODEL
```

In [272]:
```
from sklearn.svm import SVR
```

In [273]:
```
svr_model = SVR(kernel='rbf', C=1.0, epsilon=0.2)
```

In [274]:
```
svr_model.fit(X_train, y_train)
```

Out[274]:
```
SVR(epsilon=0.2)
```

In [275]:
```
y_preds = svr_model.predict(X_test)
```

In [276]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, y_preds)
print(mae)
```

```
1.9065024535603607
```

In [ ]:
```

```

In [277]:
```
from sklearn.metrics import mean_squared_error
```

In [278]:
```
mse = mean_squared_error(y_test, y_pred)
```

In [279]:
```
mse
```

Out[279]:
```
9.088370124113476
```

In [280]:
```
from sklearn.metrics import r2_score

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

```
0.9626018019210928
```

In [281]:
```
## RIDGE REGRESSION
```

In [282]:
```
from sklearn.linear_model import Ridge
```

In [283]:
```
alpha = 1.0  # Regularization strength (adjust as needed)
ridge_model = Ridge(alpha=alpha)
```

In [284]:
```
ridge_model.fit(X_train, y_train)
```

Out[284]:
```
Ridge()
```

In [285]:
```
y_predr = ridge_model.predict(X_test)
```

In [286]:
```
mse = mean_squared_error(y_test, y_predr)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 29.93273307983546
```

In [287]:
```
from sklearn.metrics import r2_score

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

```
0.9071215211162965
```

In [288]:
```
##XGBOOST
```

In [289]:
```
import xgboost as xgb
```

In [290]:
```
xg_reg = xgb.XGBRegressor(objective='reg:squarederror', colsample_bytree=0.3,
                          max_depth=5, alpha=10, n_estimators=200)
```

In [291]:
```
xg_reg.fit(X_train, y_train)
```

Out[291]:
```
XGBRegressor(alpha=10, base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.3, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=5, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=200, n_jobs=None, num_parallel_tree=None,
             predictor=None, ...)
```

In [292]:
```
y_predx = xg_reg.predict(X_test)
```

In [293]:
```
from sklearn.metrics import r2_score

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

```
0.9807711246831529
```

In [294]:
```
mse = mean_squared_error(y_test, y_predx)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 6.197052311820465
```

In [ ]:

In [ ]:

In [295]:
```
##Random Forest
```

In [296]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [297]:
```
rf_reg = RandomForestRegressor(n_estimators=300, random_state=20)
```

In [311]:
```
rf_reg.fit(X_train, y_train.ravel())
```

Out[311]:
```
RandomForestRegressor(n_estimators=300, random_state=20)
```

In [312]:
```
y_predf = rf_reg.predict(X_test)
```

In [313]:
```
from sklearn.metrics import r2_score

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

```
0.9999542614187552
```

In [314]:
```
import pickle
```

In [315]:
```
filename = "secondimp.pkl"
```

In [316]:
```
pickle.dump(rf_reg,open(filename,'wb'))
```

In [317]:
```
rf_reg1=pickle.load(open('secondimp.pkl','rb'))
```

In [318]:
```
print(rf_reg1.predict([[365,2546,0.14,1,7]]))
```

```
[96.72]

C:\Users\hp\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X d
oes not have valid feature names, but RandomForestRegressor was fitted with f
eature names
  warnings.warn(
```

In [ ]:
```

```