In [39]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:
```
df=pd.read_csv("C://Users//user//Downloads//Spotify_data.csv")
df
```

Out[2]:

	Unnamed: 0	Track Name	Artists	Album Name	Album ID	
0	0	Not Like Us	Kendrick Lamar	Not Like Us	5JjnoGJyOxfSZUZtk2rRwZ	6AI3ezQ4o3HU
1	1	Houdini	Eminem	Houdini	6Xuu2z00jxRPZei4IJ9neK	2HYFX63wP3c
2	2	BAND4BAND (feat. Lil Baby)	Central Cee, Lil Baby	BAND4BAND (feat. Lil Baby)	4AzPr5SUpNF553eC1d3aRy	7iabz12vAuVC
3	3	I Don't Wanna Wait	David Guetta, OneRepublic	I Don't Wanna Wait	0wCLHkBRKcndhMQQpeo8Ji	331I3xABO0H
4	4	Pedro	Jaxomy, Agatino Romero, Raffaella Carrà	Pedro	5y6RXjI5VPR0RyInghTbf1	48IxT5qJF0y
...	...	...	...	...	...	
222	222	Tu Chahiye	Pritam, Atif Aslam	Bajrangi Bhaijaan	4nZOPP0atfJbBIkedLYi7t	3aaiAWCet6
223	223	Aabaad Barbaad (From "Ludo")	Pritam, Arijit Singh	Aabaad Barbaad (From "Ludo")	1PzsfgcbPbiW7uflc9Zi5Z	0hFUtSsV2itY
224	224	Jag Ghoomeya	Vishal-Shekhar, Rahat Fateh Ali Khan, Irshad K...	Sultan	0tAi6H8acUKefYMIEuxcMA	4KCbZcshgib
225	225	Tumhe Kitna Pyaar Karte (From "Bawaal")	Mithoon, Arijit Singh, Manoj Muntashir	Tumhe Kitna Pyaar Karte (From "Bawaal")	20zQZcEhMLsDUn1LhPCEFY	03hJuEQpEQE
226	226	Bekhayali	Sachet Tandon	Kabir Singh	3uuu6u13U0KeVQsZ3CZKK4	4yMbbysIdI7E

227 rows × 22 columns

In [3]:
```
df.tail()
```

Out[3]:

	Unnamed: 0	Track Name	Artists	Album Name	Album ID	Trac
222	222	Tu Chahiye	Pritam, Atif Aslam	Bajrangi Bhaijaan	4nZOPP0atfJbBIkedLYi7t	3aaiAWCet6sbfOfLSn
223	223	Aabaad Barbaad (From "Ludo")	Pritam, Arijit Singh	Aabaad Barbaad (From "Ludo")	1PzsfgcbPbiW7uflc9Zi5Z	0hFUtSsV2itYEUTZGj6
224	224	Jag Ghoomeya	Vishal-Shekhar, Rahat Fateh Ali Khan, Irshad K...	Sultan	0tAi6H8acUKefYMIEuxcMA	4KCbZcshgibfJSCAkg8
225	225	Tumhe Kitna Pyaar Karte (From "Bawaal")	Mithoon, Arijit Singh, Manoj Muntashir	Tumhe Kitna Pyaar Karte (From "Bawaal")	20zQZcEhMLsDUn1LhPCEFY	03hJuEQpEQERrHpjcXK
226	226	Bekhayali	Sachet Tandon	Kabir Singh	3uuu6u13U0KeVQsZ3CZKK4	4yMbbysIdI7E3Wgiaugr

5 rows × 22 columns

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

Out[4]:

	Unnamed: 0	Track Name	Artists	Album Name	Album ID	
0	0	Not Like Us	Kendrick Lamar	Not Like Us	5JjnoGJyOxfSZUZtk2rRwZ	6AI3ezQ4o3HUoP
1	1	Houdini	Eminem	Houdini	6Xuu2z00jxRPZei4IJ9neK	2HYFX63wP3otVI
2	2	BAND4BAND (feat. Lil Baby)	Central Cee, Lil Baby	BAND4BAND (feat. Lil Baby)	4AzPr5SUpNF553eC1d3aRy	7iabz12vAuVQYy
3	3	I Don't Wanna Wait	David Guetta, OneRepublic	I Don't Wanna Wait	0wCLHkBRKcndhMQQpeo8Ji	331I3xABO0HMr1
4	4	Pedro	Jaxomy, Agatino Romero, Raffaella Carrà	Pedro	5y6RXjI5VPR0RyInghTbf1	48IxT5qJF0yYyf2

5 rows × 22 columns

In [5]:
```
df.shape
```

Out[5]:
```
(227, 22)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Unnamed: 0        227 non-null    int64  
 1   Track Name        227 non-null    object 
 2   Artists           227 non-null    object 
 3   Album Name        227 non-null    object 
 4   Album ID          227 non-null    object 
 5   Track ID          227 non-null    object 
 6   Popularity        227 non-null    int64  
 7   Release Date      227 non-null    object 
 8   Duration (ms)     227 non-null    int64  
 9   Explicit          227 non-null    bool   
 10  External URLs     227 non-null    object 
 11  Danceability      227 non-null    float64
 12  Energy            227 non-null    float64
 13  Key               227 non-null    int64  
 14  Loudness          227 non-null    float64
 15  Mode              227 non-null    int64  
 16  Speechiness       227 non-null    float64
 17  Acousticness      227 non-null    float64
 18  Instrumentalness  227 non-null    float64
 19  Liveness          227 non-null    float64
 20  Valence           227 non-null    float64
 21  Tempo             227 non-null    float64
dtypes: bool(1), float64(9), int64(5), object(7)
memory usage: 37.6+ KB
```

In [7]:
```
df.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	
Unnamed: 0	227.0	113.000000	65.673435	0.000000	56.5000	113.000000	
Popularity	227.0	71.850220	10.241100	13.000000	68.0000	72.000000	
Duration (ms)	227.0	219254.881057	60483.492317	96947.000000	170554.5000	222462.000000	2
Danceability	227.0	0.635639	0.155123	0.271000	0.5520	0.634000	
Energy	227.0	0.646665	0.159150	0.236000	0.5395	0.655000	
Key	227.0	5.458150	3.760738	0.000000	2.0000	6.000000	
Loudness	227.0	-6.516670	2.099543	-15.073000	-7.8300	-6.346000	
Mode	227.0	0.678414	0.468117	0.000000	0.0000	1.000000	
Speechiness	227.0	0.079576	0.085100	0.024600	0.0338	0.042100	
Acousticness	227.0	0.375060	0.300084	0.000307	0.0650	0.393000	
Instrumentalness	227.0	0.028890	0.137225	0.000000	0.0000	0.000002	
Liveness	227.0	0.177797	0.121366	0.029700	0.1010	0.127000	
Valence	227.0	0.472441	0.193902	0.038500	0.3245	0.462000	
Tempo	227.0	119.466361	26.154889	61.311000	95.4575	122.925000	

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Unnamed: 0', 'Track Name', 'Artists', 'Album Name', 'Album ID',
       'Track ID', 'Popularity', 'Release Date', 'Duration (ms)', 'Explicit',
       'External URLs', 'Danceability', 'Energy', 'Key', 'Loudness', 'Mode',
       'Speechiness', 'Acousticness', 'Instrumentalness', 'Liveness',
       'Valence', 'Tempo'],
      dtype='object')
```

In [9]:
```
df.drop(columns=['Unnamed: 0','Track Name','Album ID','Track ID','Explicit'],axis
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Artists             0
Album Name          0
Popularity          0
Release Date        0
Duration (ms)       0
External URLs       0
Danceability        0
Energy              0
Key                 0
Loudness            0
Mode                0
Speechiness         0
Acousticness        0
Instrumentalness    0
Liveness            0
Valence             0
Tempo               0
dtype: int64
```

In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
15
```

In [12]:
```
df.drop_duplicates()
```

Out[12]:

Artists	Album Name	Popularity	Release Date	Duration (ms)	External URLs
Kendrick Lamar	Not Like Us	96	2024-05-04	274192	https://open.spotify.com/track/6AI3ezQ4o3HUoP6...
Eminem	Houdini	94	2024-05-31	227239	https://open.spotify.com/track/2HYFX63wP3otVIv...
entral Cee, Lil Baby	BAND4BAND (feat. Lil Baby)	91	2024-05-23	140733	https://open.spotify.com/track/7iabz12vAuVQYye...
David Guetta, neRepublic	I Don't Wanna Wait	90	2024-04-05	149668	https://open.spotify.com/track/331I3xABO0HMr1K...
Jaxomy, Agatino Romero, Raffaella Carrà	Pedro	89	2024-03-29	144846	https://open.spotify.com/track/48IxT5qJF0yYyf2...
...	...	...	...	...	...
Pritam, Atif Aslam	Bajrangi Bhaijaan	66	2015-07-07	272680	https://open.spotify.com/track/3aaiAWCet6sbfOf...
ritam, Arijit Singh	Aabaad Barbaad (From "Ludo")	58	2020-10-21	309103	https://open.spotify.com/track/0hFUtSsV2itYEUT...
Vishal-Shekhar, ahat Fateh Ali Khan, Irshad K...	Sultan	62	2016-05-31	281992	https://open.spotify.com/track/4KCbZcshgibfJSC...
Mithoon, Arijit Singh, Manoj Muntashir	Tumhe Kitna Pyaar Karte (From "Bawaal")	65	2023-07-07	305232	https://open.spotify.com/track/03hJuEQpEQERrHp...
Sachet Tandon	Kabir Singh	61	2019-06-14	371791	https://open.spotify.com/track/4yMbbysIdI7E3Wg...

s × 17 columns

In [13]:
```
df.drop('External URLs',axis=1,inplace=True)
```

In [16]:
```
df
```

Out[16]:

	Artists	Album Name	Popularity	Release Date	Duration (ms)	Danceability	Energy	Key	Loudness
0	Kendrick Lamar	Not Like Us	96	2024-05-04	274192	0.898	0.472	1	-7.001
1	Eminem	Houdini	94	2024-05-31	227239	0.936	0.887	9	-2.760
2	Central Cee, Lil Baby	BAND4BAND (feat. Lil Baby)	91	2024-05-23	140733	0.882	0.764	11	-5.241
3	David Guetta, OneRepublic	I Don't Wanna Wait	90	2024-04-05	149668	0.681	0.714	1	-4.617
4	Jaxomy, Agatino Romero, Raffaella Carrà	Pedro	89	2024-03-29	144846	0.788	0.936	9	-6.294
...	...	...	...	...	...	...	...	...	...
222	Pritam, Atif Aslam	Bajrangi Bhaijaan	66	2015-07-07	272680	0.565	0.744	7	-5.817
223	Pritam, Arijit Singh	Aabaad Barbaad (From "Ludo")	58	2020-10-21	309103	0.626	0.522	7	-5.857
224	Vishal-Shekhar, Rahat Fateh Ali Khan, Irshad K...	Sultan	62	2016-05-31	281992	0.484	0.565	11	-7.954
225	Mithoon, Arijit Singh, Manoj Muntashir	Tumhe Kitna Pyaar Karte (From "Bawaal")	65	2023-07-07	305232	0.602	0.374	10	-9.849
226	Sachet Tandon	Kabir Singh	61	2019-06-14	371791	0.296	0.582	9	-5.180

227 rows × 16 columns

In [29]:
```
num_col=df.select_dtypes(include=["float64","int64"])
num_col
```

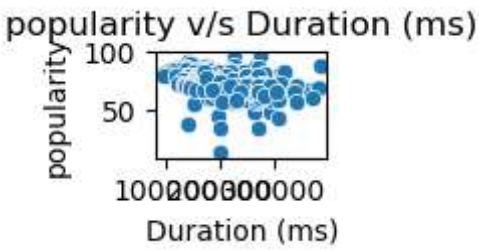
Out[29]:

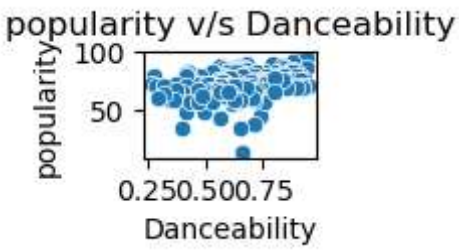
	Popularity	Duration (ms)	Danceability	Energy	Key	Loudness	Mode	Speechiness	Acousticness
0	96	274192	0.898	0.472	1	-7.001	1	0.0776	0.0107
1	94	227239	0.936	0.887	9	-2.760	0	0.0683	0.0292
2	91	140733	0.882	0.764	11	-5.241	1	0.2040	0.3590
3	90	149668	0.681	0.714	1	-4.617	0	0.0309	0.0375
4	89	144846	0.788	0.936	9	-6.294	1	0.3010	0.0229
...	...	...	...	...	...	...	...	...	...
222	66	272680	0.565	0.744	7	-5.817	1	0.0446	0.4030
223	58	309103	0.626	0.522	7	-5.857	1	0.0317	0.6860
224	62	281992	0.484	0.565	11	-7.954	1	0.0347	0.4790
225	65	305232	0.602	0.374	10	-9.849	0	0.0328	0.9240
226	61	371791	0.296	0.582	9	-5.180	0	0.0413	0.4490

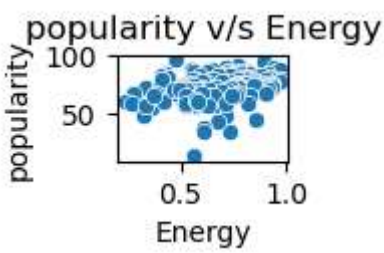
227 rows × 13 columns

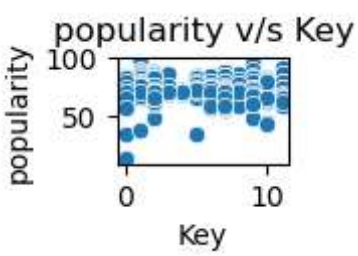
In [56]:

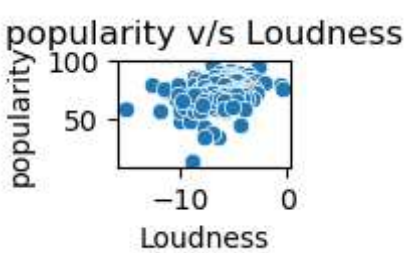
```
for i, col in enumerate(num_col):
    if col!="Popularity":
        plt.subplot(6,5,i)
        sns.scatterplot(x=col,y="Popularity",data=df)
        plt.title(f"popularity v/s {col}")
        plt.xlabel(col)
        plt.ylabel("popularity")
        plt.tight_layout
        plt.show()
```

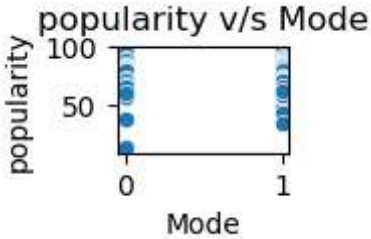


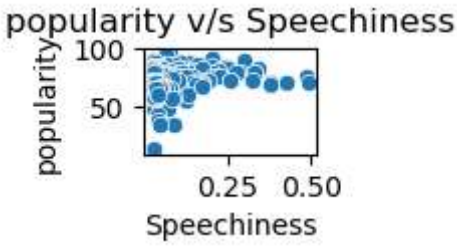


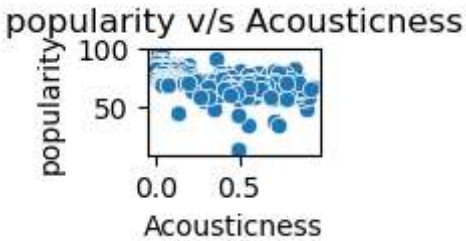


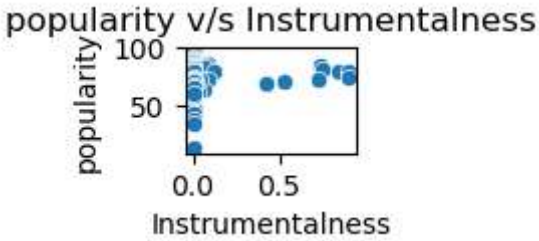


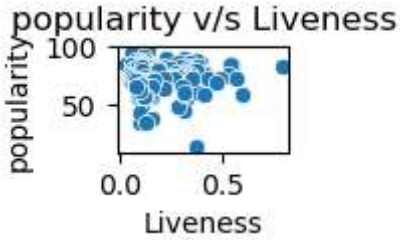


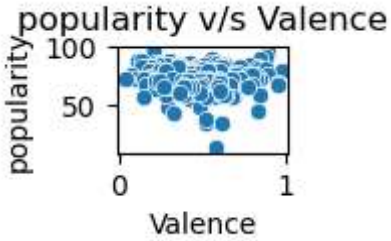

popularity v/s Mode
popularity
100
50
0
1
Mode

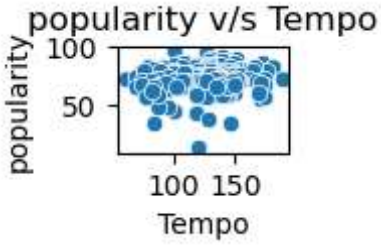

popularity v/s Speechiness
popularity
100
50
0.25
0.50
Speechiness


popularity v/s Acousticness
popularity
100
50
0.0
0.5
Acousticness


popularity v/s Instrumentalness
popularity
100
50
0.0
0.5
Instrumentalness

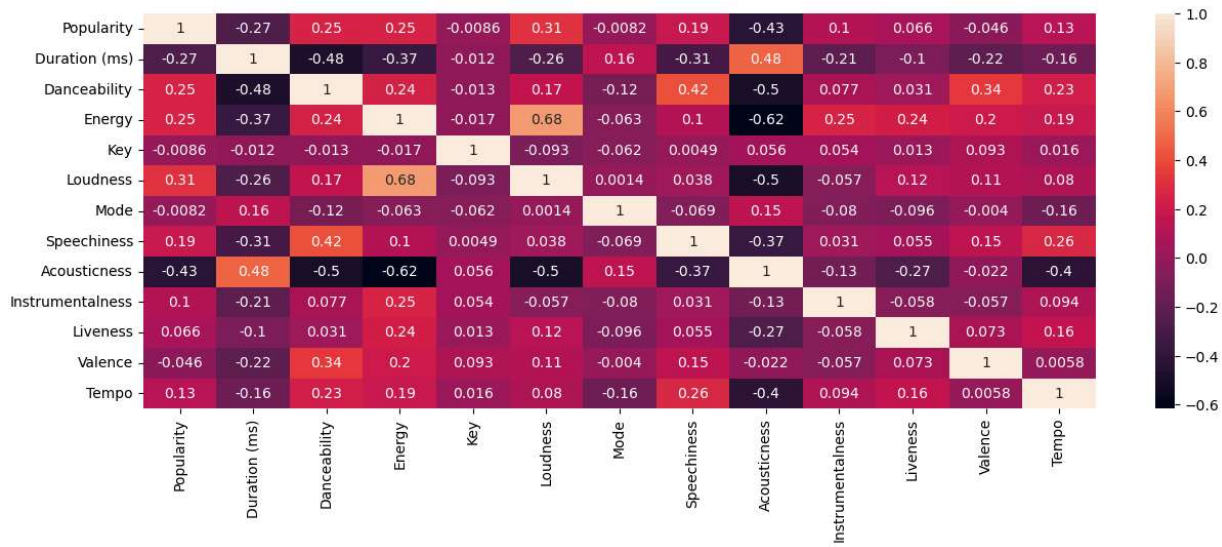

popularity v/s Liveness
popularity
100
50
0.0
0.5
Liveness


popularity v/s Valence
popularity
100
50
0
1
Valence

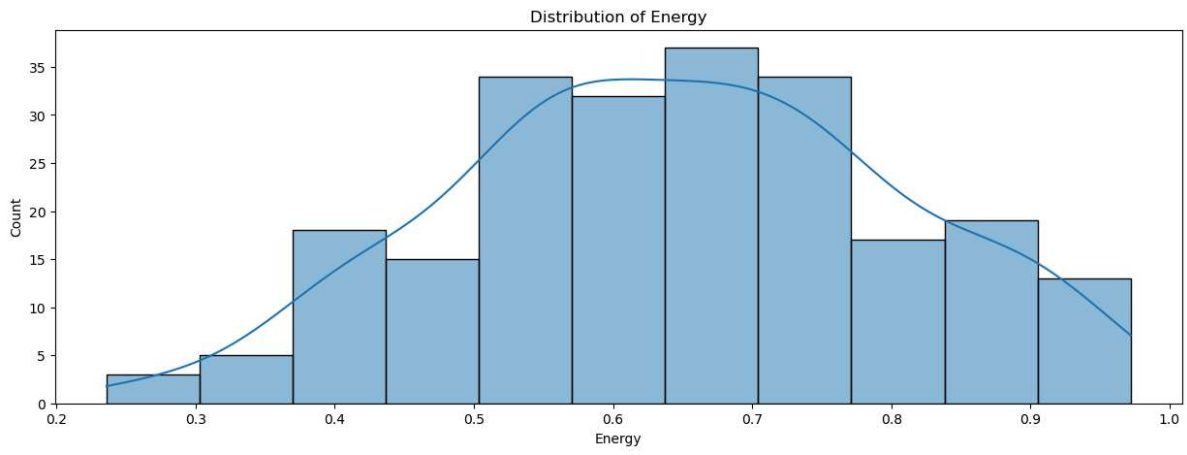


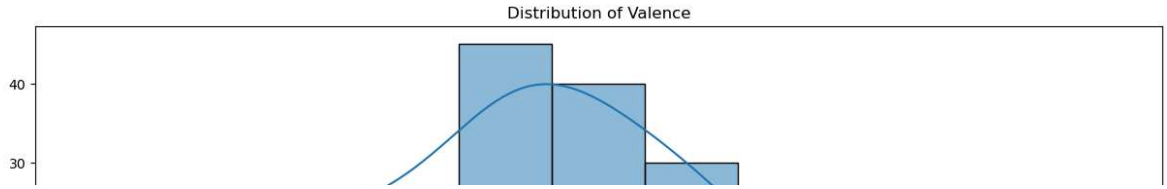
In [35]:
```
corr=num_col.corr()
plt.figure(figsize=(15,5))
sns.heatmap(corr,annot=True)
```

Out[35]: <Axes: >



In [36]:
```
features=['Energy', 'Valence', 'Danceability', 'Loudness', 'Acousticness']
for feature in features:
    plt.figure(figsize=(15,5))
    sns.histplot(df[feature],kde=True)
    plt.title(f"Distribution of {feature}")
    plt.show()
```





In [38]:
```
x=df[features]
y=df["Popularity"]
```

In [40]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
```

In [42]:
```
scaler=StandardScaler()
scaler.fit_transform(x_train)
```

```
       [ 6.36992540e-01, -2.82905370e-01, -4.04520988e-01,
         3.28873590e-01,  1.16901117e-01],
       [ 1.82967515e+00, -1.46225205e+00, -4.89251899e-01,
         3.31238402e-01, -9.68959408e-01],
       [-1.17896795e+00, -1.62422238e+00, -1.58621196e+00,
        -2.18817035e-01,  2.85278224e-01],
       [-1.15951552e+00, -1.56348351e+00, -1.67970287e+00,
        -2.17871110e-01,  2.50915549e-01],
       [-4.65712044e-01, -1.76612540e-01, -3.89528257e-01,
        -5.98389074e-03,  6.97630322e-01],
       [ 1.77131785e+00,  1.90875549e+00,  1.18735182e+00,
         2.07741611e+00,  4.60527866e-01],
       [-1.54473103e-01, -1.26991228e+00,  1.29330819e+00,
        -1.10981575e-01, -8.58998848e-01],
       [-3.12743550e-02, -1.63940710e+00, -2.89804616e-01,
        -2.93929758e+00,  1.95935270e-01],
       [ 1.51843621e+00, -3.99500719e-02,  5.01751789e-01,
         1.73972086e+00, -1.07479645e+00],
       [ 1.62218252e+00, -3.43644445e-01,  1.90115409e-01,
         8.69942739e-01, -1.15039433e+00],
```

In [43]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score,mean_absolute_error
```

In [44]:
```
models={
    "lr":LinearRegression(),
    "rf":RandomForestRegressor(),
    "dt":DecisionTreeRegressor(),
    "svr":SVR()
}
```

In [45]:
```
results = []

for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    results.append({
        'Model': name,
        'MSE': mse,
        'MAE': mae,
        'R2 Score': r2
    })
```

In [47]:
```
result_df=pd.DataFrame(results)
result_df
```

Out[47]:

	Model	MSE	MAE	R2 Score
0	lr	65.680843	6.093160	0.013202
1	rf	45.287017	5.412899	0.319601
2	dt	78.630435	6.239130	-0.181355
3	svr	60.172709	5.681649	0.095957

In [ ]:
```
🔍 Insights
Random Forest Regressor performs best overall, with the lowest MSE and highest R²

SVR is second-best, but the performance gain over Linear Regression is small.

Linear Regression and Decision Tree perform poorly — especially the Decision Tree
```

In [48]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
```

In [49]:
```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

In [51]:
```
rf = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,                       # 5-fold cross-validation
    n_jobs=-1,                  # Use all CPU cores
    verbose=2,
    scoring='r2'
)
```

In [52]:
```
grid_search.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
```

Out[52]:

```
▸ GridSearchCV
  (https://scikit-learn.org/1.6/modules/generated/sklearn.model_selec
▸ best_estimator_: RandomForestRegressor
  ▸ RandomForestRegressor
    (https://scikit-learn.org/1.6/modules/generated/sklearn.ensemble.RandomFor
```

In [53]:
```
print("Best Parameters:", grid_search.best_params_)

best_rf = grid_search.best_estimator_
y_pred = best_rf.predict(x_test)

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

print("MSE:", mean_squared_error(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split':
2, 'n_estimators': 100}
MSE: 51.77891775548146
MAE: 5.93084533730747
R² Score: 0.22206624559359012
```

In [ ]: