In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.svm import LinearSVR
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

from sklearn.model_selection import GridSearchCV
```

In [5]:

```python
df = pd.read_csv(r"C:\Users\91797\Downloads\archive (2)\IMDb Movies India.csv", encoding='ISO-8859-1')
```

In [6]:

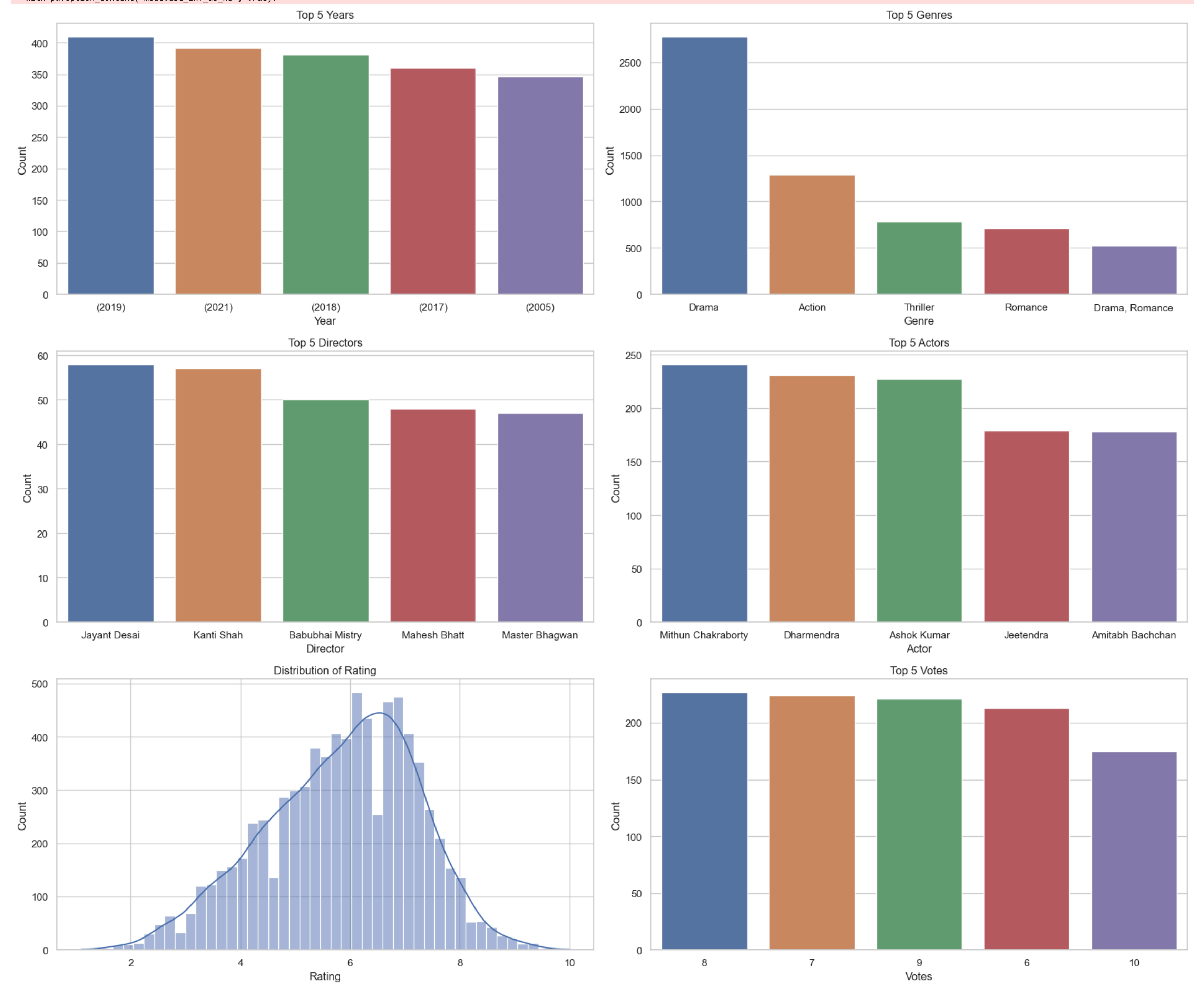
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      15509 non-null  object
 1   Year      14981 non-null  object
 2   Duration  7240 non-null   object
 3   Genre     13632 non-null  object
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   object
 6   Director  14984 non-null  object
 7   Actor 1   13892 non-null  object
 8   Actor 2   13125 non-null  object
 9   Actor 3   12365 non-null  object
dtypes: float64(1), object(9)
memory usage: 1.2+ MB
```

In [7]:

```python
sns.set(style="whitegrid")

# Create a figure and a set of subplots
fig, axes = plt.subplots(3, 2, figsize=(18, 15))

# Plot top 5 Years
top_5_years = df['Year'].value_counts().nlargest(5)
sns.barplot(x=top_5_years.index, y=top_5_years.values, ax=axes[0, 0])
axes[0, 0].set_title('Top 5 Years')
axes[0, 0].set_xlabel('Year')
axes[0, 0].set_ylabel('Count')

# Plot top 5 Genres
top_5_genres = df['Genre'].value_counts().nlargest(5)
sns.barplot(x=top_5_genres.index, y=top_5_genres.values, ax=axes[0, 1])
axes[0, 1].set_title('Top 5 Genres')
axes[0, 1].set_xlabel('Genre')
axes[0, 1].set_ylabel('Count')

# Plot top 5 Directors
top_5_directors = df['Director'].value_counts().nlargest(5)
sns.barplot(x=top_5_directors.index, y=top_5_directors.values, ax=axes[1, 0])
axes[1, 0].set_title('Top 5 Directors')
axes[1, 0].set_xlabel('Director')
axes[1, 0].set_ylabel('Count')

# Combine Actor 1, Actor 2, Actor 3 and plot top 5 Actors
all_actors = pd.concat([df['Actor 1'], df['Actor 2'], df['Actor 3']])
top_5_actors = all_actors.value_counts().nlargest(5)
sns.barplot(x=top_5_actors.index, y=top_5_actors.values, ax=axes[1, 1])
axes[1, 1].set_title('Top 5 Actors')
axes[1, 1].set_xlabel('Actor')
axes[1, 1].set_ylabel('Count')

# Plot distribution of Rating
sns.histplot(df['Rating'].dropna(), kde=True, ax=axes[2, 0])
axes[2, 0].set_title('Distribution of Rating')
axes[2, 0].set_xlabel('Rating')
axes[2, 0].set_ylabel('Count')

# Plot top 5 Votes
top_5_votes = df['Votes'].value_counts().nlargest(5)
sns.barplot(x=top_5_votes.index, y=top_5_votes.values, ax=axes[2, 1])
axes[2, 1].set_title('Top 5 Votes')
axes[2, 1].set_xlabel('Votes')
axes[2, 1].set_ylabel('Count')

# Adjust the layout
plt.tight_layout()
plt.show()
```

```
C:\Users\91797\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [8]:

```python
missing_values = df.isnull().sum()
print(missing_values)
```

```
Name           0
Year         528
Duration    8269
Genre       1877
Rating      7590
Votes       7589
Director     525
Actor 1     1617
Actor 2     2384
Actor 3     3144
dtype: int64
```

In [9]:

```python
df.head()
```

Out[9]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birbal	Rajendra Bhatia
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid
2	#Homecoming	(2021)	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur	Roy Angana
3	#Yaaram	(2019)	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
4	...And Once Again	(2010)	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Rituparna Sengupta	Antara Mali

In [10]:

```python
df.describe()
```

Out[10]:

	Rating
count	7919.000000
mean	5.841621
std	1.381777
min	1.100000
25%	4.900000
50%	6.000000
75%	6.800000
max	10.000000

In [11]:

```python
# Drop rows with missing values in Rating and Votes
df.dropna(subset=['Rating', 'Votes'], inplace=True)

# Convert Rating to numeric
df['Votes'] = df['Votes'].str.replace(',', '').astype(int)

# Convert Rating to numeric
df['Year'] = pd.to_numeric(df['Year'], errors='coerce')

# Convert Duration to numeric
df['Duration'] = df['Duration'].str.replace(' min', '', regex=False)
df['Duration'] = pd.to_numeric(df['Duration'], errors='coerce')
```

In [12]:

```python
categorical_cols = ['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']
for col in categorical_cols:
    df.loc[:, col] = df[col].fillna('Unknown')
```

In [13]:

```python
max_categories = 20  # Maximum number of categories to keep in each column
for col in categorical_cols:
    top_categories = df[col].value_counts().nlargest(max_categories).index
    df[col] = df[col].where(df[col].isin(top_categories), 'Other')
```

In [14]:

```python
encoder = OneHotEncoder(sparse_output=True, handle_unknown='ignore')
encoded_features = encoder.fit_transform(df[['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']])
```

In [15]:

```python
print(f"Shape of encoded_features: {encoded_features.shape}")
feature_names = encoder.get_feature_names_out(categorical_cols)
print(f"Length of feature_names: {len(feature_names)}")
```

```
Shape of encoded_features: (7919, 105)
Length of feature_names: 105
```

In [16]:

```python
# Create a DataFrame from the encoded features
encoded_df = pd.DataFrame(encoded_features.toarray(), index=df.index, columns=feature_names)
df = pd.concat([df, encoded_df], axis=1)
```

In [17]:

```python
features = encoded_df.columns
target = 'Rating'
```

In [18]:

```python
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)
```

In [19]:

```python
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest Regression': RandomForestRegressor(random_state=42),
    'Gradient Boosting Regression': GradientBoostingRegressor(random_state=42),
    'Linear SVR': LinearSVR(dual='auto', random_state=42)
}

results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    results[name] = {'MAE': mae, 'MSE': mse, 'RMSE': rmse}
```

```
---------------------------------------------------------------------------
InvalidParameterError                     Traceback (most recent call last)
Cell In[19], line 10
      8 results = {}
      9 for name, model in models.items():
---> 10     model.fit(X_train, y_train)
     11     y_pred = model.predict(X_test)
     12     mae = mean_absolute_error(y_test, y_pred)

File ~\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:507, in LinearSVR.fit(self, X, y, sample_weight)
    483 def fit(self, X, y, sample_weight=None):
    484     """Fit the model according to the given training data.
    485 
    486     Parameters
   (...)
    505         An instance of the estimator.
    506     """
--> 507     self._validate_params()
    509     X, y = self._validate_data(
    510         X,
    511         y,
   (...)
    515         accept_large_sparse=False,
    516     )
    517     penalty = "l2"  # SVR only accepts l2 penalty

File ~\anaconda3\Lib\site-packages\sklearn\base.py:600, in BaseEstimator._validate_params(self)
    592 def _validate_params(self):
    593     """Validate types and values of constructor parameters
    594 
    595     The expected type and values must be defined in the `_parameter_constraints`
   (...)
    598     accepted constraints.
    599     """
--> 600     validate_parameter_constraints(
    601         self._parameter_constraints,
    602         self.get_params(deep=False),
    603         caller_name=self.__class__.__name__,
    604     )

File ~\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py:97, in validate_parameter_constraints(parameter_constraints, params, caller_name)
     91 else:
     92     constraints_str = (
     93         f"{', '.join([str(c) for c in constraints[:-1]])} or"
     94         f" {constraints[-1]}"
     95     )
---> 97 raise InvalidParameterError(
     98     f"The {param_name!r} parameter of {caller_name} must be"
     99     f" {constraints_str}. Got {param_val!r} instead."
    100 )

InvalidParameterError: The 'dual' parameter of LinearSVR must be an instance of 'bool', an instance of 'numpy.bool_' or an instance of 'int'. Got 'auto' instead.
```

In [20]:

```python
# Displaying the results
for name, metrics in results.items():
    print(f"Model: {name}")
    for metric, value in metrics.items():
        print(f"{metric}: {value}")
    print("-" * 30)
```

```
Model: Linear Regression
MAE: 1.0068999088171757
MSE: 1.596590165857489
RMSE: 1.263562489890187
------------------------------
Model: Random Forest Regression
MAE: 1.012252347262429
MSE: 1.6161374271593203
RMSE: 1.2712739386769951
------------------------------
Model: Gradient Boosting Regression
MAE: 1.019832686684646
MSE: 1.621925766501472
RMSE: 1.273548493973226
------------------------------
```

In [21]:

```python
param_grid = {
    'C': [0.1, 1, 10, 100],
    'epsilon': [0, 0.1, 0.2],
    'loss': ['epsilon_insensitive', 'squared_epsilon_insensitive']
}

linear_svr = LinearSVR(dual='auto', random_state=42)
grid_search = GridSearchCV(linear_svr, param_grid, cv=5, scoring='neg_mean_absolute_error')
grid_search.fit(X_train, y_train)

print("Hyperparameter terbaik:", grid_search.best_params_)

best_linear_svr = grid_search.best_estimator_
y_pred = best_linear_svr.predict(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[21], line 9
      7 linear_svr = LinearSVR(dual='auto', random_state=42)
      8 grid_search = GridSearchCV(linear_svr, param_grid, cv=5, scoring='neg_mean_absolute_error')
----> 9 grid_search.fit(X_train, y_train)
     11 print("Hyperparameter terbaik:", grid_search.best_params_)
     13 best_linear_svr = grid_search.best_estimator_

File ~\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:874, in BaseSearchCV.fit(self, X, y, groups, **fit_params)
    868     results = self._format_results(
    869         all_candidate_params, n_splits, all_out, all_more_results
    870     )
    872     return results
--> 874 self._run_search(evaluate_candidates)
    876 # multimetric is determined here because in the case of a callable
    877 # self.scoring the return type is only known after calling
    878 first_test_score = all_out[0]["test_scores"]

File ~\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:1388, in GridSearchCV._run_search(self, evaluate_candidates)
   1386 def _run_search(self, evaluate_candidates):
   1387     """Search all candidates in param_grid"""
-> 1388     evaluate_candidates(ParameterGrid(self.param_grid))

File ~\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:851, in BaseSearchCV.fit.<locals>.evaluate_candidates(candidate_params, cv, more_results)
    844 elif len(out) != n_candidates * n_splits:
    845     raise ValueError(
    846         "cv.split and cv.get_n_splits returned "
    847         "inconsistent results. Expected {} "
    848         "splits, got {}".format(n_splits, len(out) // n_candidates)
    849     )
--> 851 _warn_or_raise_about_fit_failures(out, self.error_score)
    853 # For callable self.scoring, the return type is only know after
    854 # calling. If the return type is a dictionary, the error scores
    855 # can now be inserted with the correct key. The type checking
    856 # of out will be done in `_insert_error_scores`.
    857 if callable(self.scoring):

File ~\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:367, in _warn_or_raise_about_fit_failures(results, error_score)
    360 if num_failed_fits == num_fits:
    361     all_fits_failed_message = (
    362         f"\nAll the {num_fits} fits failed.\n"
    363         "It is very likely that your model is misconfigured.\n"
    364         "You can try to debug the error by setting error_score='raise'.\n\n"
    365         f"Below are more details about the failures:\n{fit_errors_summary}"
    366     )
--> 367     raise ValueError(all_fits_failed_message)
    369 else:
    370     some_fits_failed_message = (
    371         f"\n{num_failed_fits} fits failed out of a total of {num_fits}.\n"
    372         "The score on these train-test partitions for these parameters"
   (...)
    376         f"Below are more details about the failures:\n{fit_errors_summary}"
    377     )

ValueError: 
All the 120 fits failed.
It is very likely that your model is misconfigured.
You can try to debug the error by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
120 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\91797\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\91797\anaconda3\Lib\site-packages\sklearn\svm\_classes.py", line 507, in fit
    self._validate_params()
  File "C:\Users\91797\anaconda3\Lib\site-packages\sklearn\base.py", line 600, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\91797\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'dual' parameter of LinearSVR must be an instance of 'bool', an instance of 'numpy.bool_' or an instance of 'int'. Got 'auto' instead.
```

In [ ]: