UNEMPLOYMENT-ANALYSIS

Aim

Analyzing the patterns and trends of unemployment in India, focusing on different regions and states. By examining historical data and identifying key contributing factors, such as economic indicators, education levels, and industry sectors, the project aims to provide insights into the socio-economic impact of unemployment and offer recommendations for policymakers and organizations to address the issue effectively. Through data visualization and analysis, the project seeks to enhance understanding and awareness of unemployment challenges and opportunities for intervention and policy development.

Procedure

Data Cleaning

Import nessesary libraries

```
import pandas as pd
```

Read the data set

```
df = pd.read_csv("/kaggle/input/unemployment-in-india/Unemployment-in-India.csv")
```

```
df.head()
```

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area
0	Andhra Pradesh	31-05-2019	Monthly	3.65	11999139.0	43.24	Rural
1	Andhra Pradesh	30-06-2019	Monthly	3.05	11755881.0	42.05	Rural

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 754 entries, 0 to 753
Data columns (total 7 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Region                                   740 non-null    object 
 1    Date                                    740 non-null    object 
 2    Frequency                               740 non-null    object 
 3    Estimated Unemployment Rate (%)         740 non-null    float64
 4    Estimated Employed                      740 non-null    float64
 5    Estimated Labour Participation Rate (%) 740 non-null    float64
 6   Area                                     740 non-null    object 
dtypes: float64(3), object(4)
memory usage: 41.4+ KB
```

```
df.describe()
```

	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)
count	740.000000	7.400000e+02	740.000000
mean	11.787946	7.204460e+06	42.630122
std	10.721298	8.087988e+06	8.111094
min	0.000000	4.942000e+04	13.330000
25%	4.657500	1.190404e+06	38.062500
50%	8.350000	4.744178e+06	41.160000
75%	15.887500	1.127549e+07	45.505000
max	76.740000	4.577751e+07	72.570000

Finding the null values

"True" indicates that the value in the original DataFrame is missing ('NaN').

"False" indicates that the value in the original DataFrame is not missing.

```
df.isnull()
```

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
749	False	False	False	False	False	False	False
750	False	False	False	False	False	False	False
751	False	False	False	False	False	False	False
752	False	False	False	False	False	False	False
753	False	False	False	False	False	False	False

Data Manipulation

Removing the null values

```
df.dropna
```

```
<bound method DataFrame.dropna of             Region         Date  Frequency   Estimated Unemployment Rate (%)  \
0    Andhra Pradesh   31-05-2019    Monthly                              3.65
1    Andhra Pradesh   30-06-2019    Monthly                              3.05
2    Andhra Pradesh   31-07-2019    Monthly                              3.75
3    Andhra Pradesh   31-08-2019    Monthly                              3.32
4    Andhra Pradesh   30-09-2019    Monthly                              5.17
..              ...          ...        ...                               ...
749     West Bengal   29-02-2020    Monthly                              7.55
750     West Bengal   31-03-2020    Monthly                              6.67
751     West Bengal   30-04-2020    Monthly                             15.63
752     West Bengal   31-05-2020    Monthly                             15.22
753     West Bengal   30-06-2020    Monthly                              9.86

      Estimated Employed   Estimated Labour Participation Rate (%)   Area
0             11999139.0                                    43.24  Rural
1             11755881.0                                    42.05  Rural
2             12086707.0                                    43.50  Rural
3             12285693.0                                    43.97  Rural
4             12256762.0                                    44.68  Rural
..                   ...                                      ...    ...
749           10871168.0                                    44.09  Urban
```

```
750           10806105.0                                43.34  Urban
751            9299466.0                                41.20  Urban
752            9240903.0                                40.67  Urban
753            9088931.0                                37.57  Urban

[754 rows x 7 columns]>
```

Removing Duplicates

```
df.drop_duplicates()
```

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area
0	Andhra Pradesh	31-05-2019	Monthly	3.65	11999139.0	43.24	Rural
1	Andhra Pradesh	30-06-2019	Monthly	3.05	11755881.0	42.05	Rural
2	Andhra Pradesh	31-07-2019	Monthly	3.75	12086707.0	43.50	Rural
3	Andhra Pradesh	31-08-2019	Monthly	3.32	12285693.0	43.97	Rural
4	Andhra Pradesh	30-09-2019	Monthly	5.17	12256762.0	44.68	Rural
...	...	...	...	...	...	...	...

```
df.columns
```

```
Index(['Region', ' Date', ' Frequency', ' Estimated Unemployment Rate (%)',
       ' Estimated Employed', ' Estimated Labour Participation Rate (%)',
       'Area'],
      dtype='object')
```

Data Visualization

To Find the insights in the data

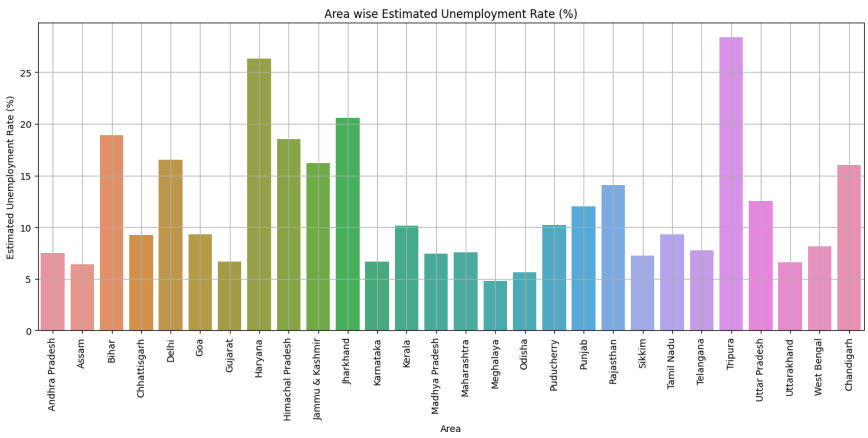
import visualization Libraries

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
print(df[' Estimated Unemployment Rate (%)'].dtype)
```

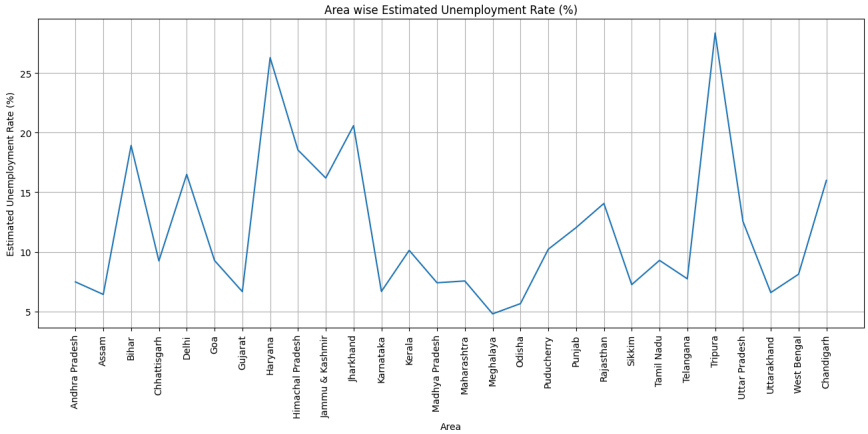
```
float64
```

```
plt.figure(figsize=(16, 6))
sns.barplot(data=df, x='Region', y=' Estimated Unemployment Rate (%)', estimator='mean', errorbar=None)
plt.title('Area wise Estimated Unemployment Rate (%)')
plt.xlabel('Area')
plt.ylabel('Estimated Unemployment Rate (%)')
plt.xticks(rotation=90)
plt.grid()
plt.show()
```

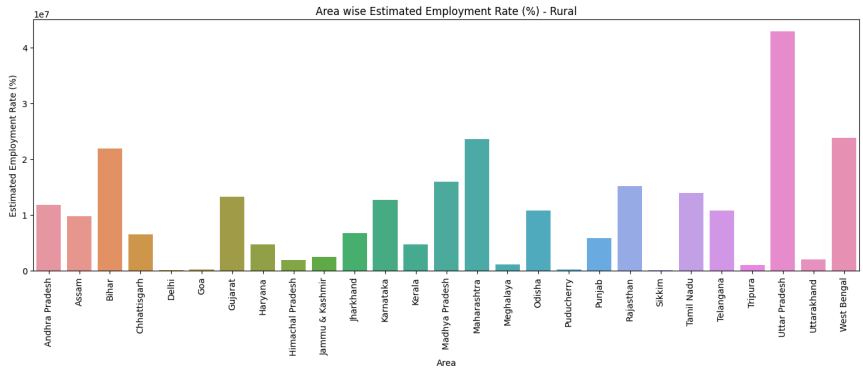


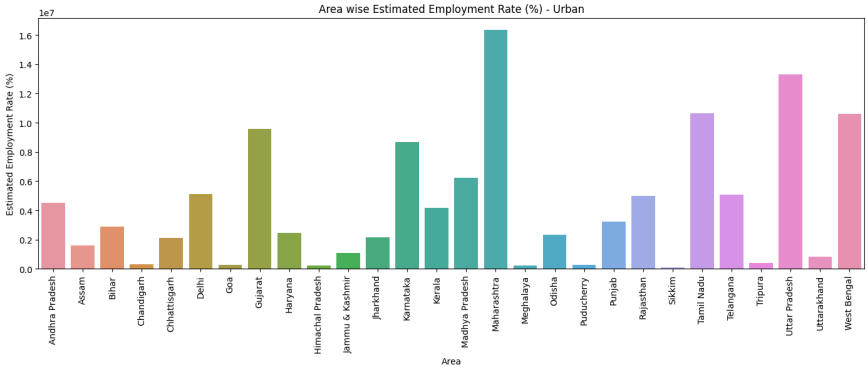

```
plt.figure(figsize=(16, 6))
sns.lineplot(data=df, x='Region', y=' Estimated Unemployment Rate (%)', estimator='average', errorbar=None)
plt.title('Area wise Estimated Unemployment Rate (%)')
plt.xlabel('Area')
plt.ylabel('Estimated Unemployment Rate (%)')
plt.xticks(rotation=90)
plt.grid()
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
```

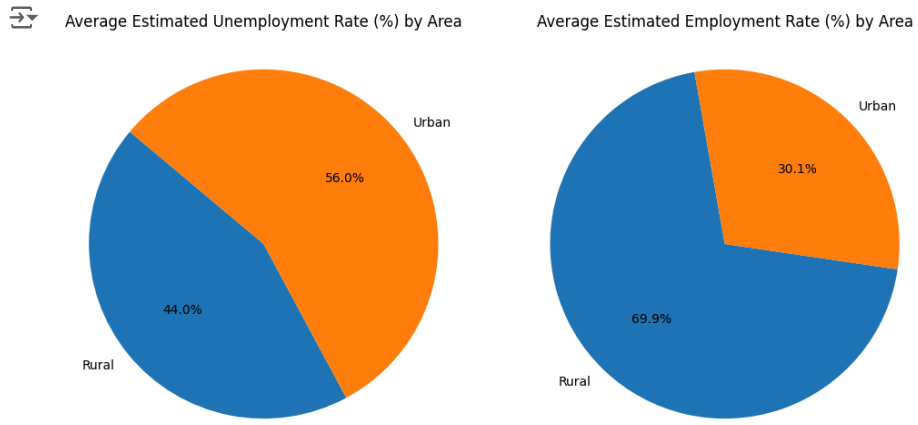



```
df_rural = df[df['Area'] == 'Rural']
df_urban = df[df['Area'] == 'Urban']
plt.figure(figsize=(14, 12))
plt.subplot(2, 1, 1)
sns.barplot(data=df_rural, x='Region', y=' Estimated Employed', estimator='mean', errorbar=None)
plt.title('Area wise Estimated Employment Rate (%) - Rural')
plt.xlabel('Area')
plt.ylabel('Estimated Employment Rate (%)')
plt.xticks(rotation=90)

plt.subplot(2, 1, 2)
sns.barplot(data=df_urban, x='Region', y=' Estimated Employed', estimator='mean', errorbar=None)
plt.title('Area wise Estimated Employment Rate (%) - Urban')
plt.xlabel('Area')
plt.ylabel('Estimated Employment Rate (%)')
plt.xticks(rotation=90)

plt.tight_layout()
plt.show()
```

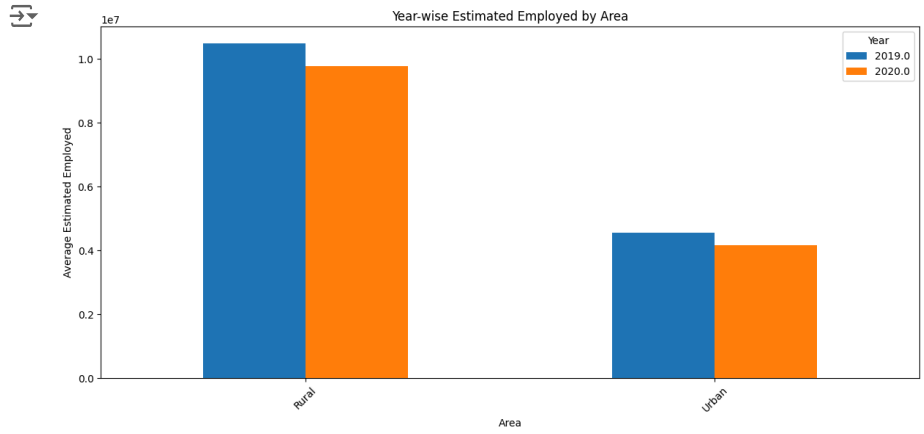





```
plt.figure(figsize=(12, 6))

# Subplot 1
plt.subplot(1, 2, 1)
region_unemployment_mean = df.groupby('Area')[' Estimated Unemployment Rate (%)'].mean()
plt.pie(region_unemployment_mean, labels=region_unemployment_mean.index, autopct='%1.1f%%', startangle=140)
plt.title('Average Estimated Unemployment Rate (%) by Area')
plt.axis('equal')

# Subplot 2
plt.subplot(1, 2, 2)
region_employment_mean = df.groupby('Area')[' Estimated Employed'].mean()
plt.pie(region_employment_mean, labels=region_employment_mean.index, autopct='%1.1f%%', startangle=100)
plt.title('Average Estimated Employment Rate (%) by Area')
plt.axis('equal')
plt.show()
```

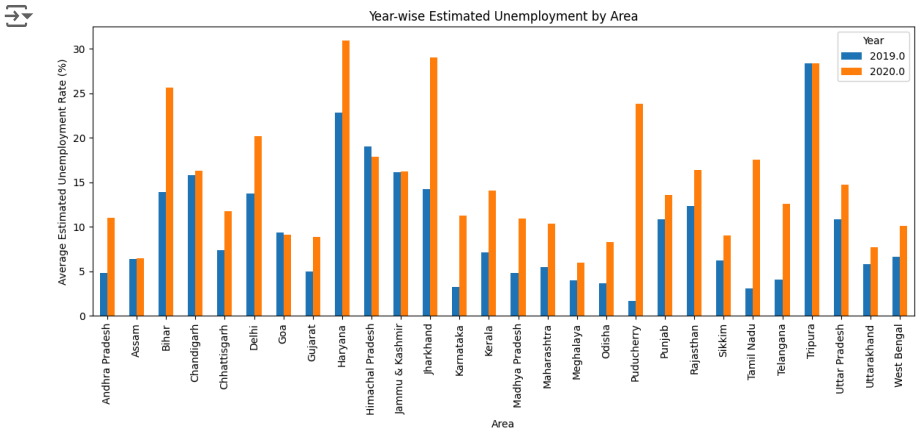



```
df[' Date'] = pd.to_datetime(df[' Date'].str.strip(), format='%d-%m-%Y')
df['Year'] = df[' Date'].dt.year
df_2019_2020 = df[df['Year'].isin([2019, 2020])]
grouped_data = df_2019_2020.groupby(['Area', 'Year'])[' Estimated Employed'].mean().unstack()

# Plotting
plt.figure(figsize=(12, 6))
grouped_data.plot(kind='bar', ax=plt.gca())
plt.title('Year-wise Estimated Employed by Area')
plt.xlabel('Area')
plt.ylabel('Average Estimated Employed')
plt.legend(title='Year')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

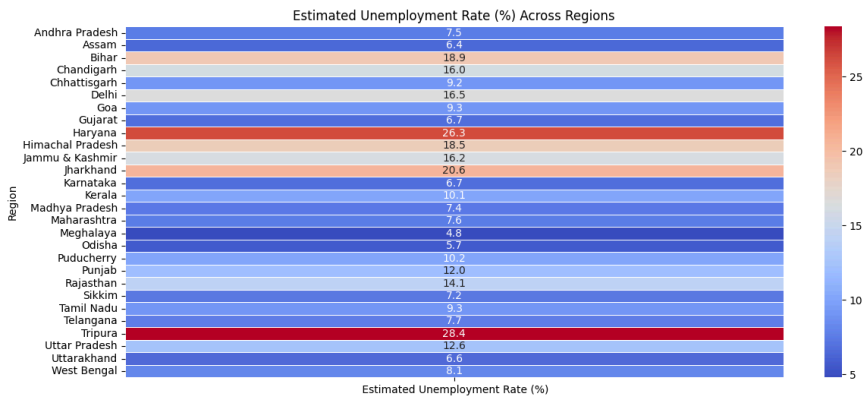


Year-wise Estimated Unemployment by Area

```
df[' Date'] = pd.to_datetime(df[' Date'], format='%d-%m-%Y')
df['Year'] = df[' Date'].dt.year
df_2019_2020 = df[df['Year'].isin([2019, 2020])]
grouped_data = df_2019_2020.groupby(['Region', 'Year'])[' Estimated Unemployment Rate (%)'].mean().unstack()

# Plotting
plt.figure(figsize=(12, 6))
grouped_data.plot(kind='bar', ax=plt.gca())
plt.title('Year-wise Estimated Unemployment by Area')
plt.xlabel('Area')
plt.ylabel('Average Estimated Unemployment Rate (%)')
plt.xticks(rotation=45)
plt.legend(title='Year')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```




```
heatmap_data = df.pivot_table(index='Region', values=' Estimated Unemployment Rate (%)', aggfunc='mean')
plt.figure(figsize=(14, 6))
sns.heatmap(heatmap_data, cmap='coolwarm', annot=True, fmt=".1f", linewidths=.5)
plt.title('Estimated Unemployment Rate (%) Across Regions')
plt.ylabel('Region')
plt.xticks(rotation=0)
plt.yticks(rotation=0)
plt.show()
```



Predicting Future Trends

Model Selection

Here we use the Linear Regression and Random Forest Regressor from the machine learning module "sklearn"

```
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score


# Convert 'Frequency' to a categorical type
df[' Frequency'] = df[' Frequency'].astype('category').cat.codes

# Check the data types
print(df.dtypes)

# Verify no non-numeric values are present in numeric columns
print(df.isnull().sum())
print(df[df.columns[df.dtypes == 'object']].head())
```

```
Region                                    object
 Date                                 datetime64[ns]
 Frequency                                 int8
 Estimated Unemployment Rate (%)         float64
 Estimated Employed                      float64
 Estimated Labour Participation Rate (%) float64
Area                                      object
Year                                     float64
dtype: object
Region                                    14
 Date                                     14
 Frequency                                 0
 Estimated Unemployment Rate (%)          14
 Estimated Employed                       14
 Estimated Labour Participation Rate (%)  14
Area                                      14
Year                                      14
dtype: int64
           Region   Area
0  Andhra Pradesh  Rural
1  Andhra Pradesh  Rural
```

```
2  Andhra Pradesh  Rural
3  Andhra Pradesh  Rural
4  Andhra Pradesh  Rural
```

```
# Select features and target variable
X = df[[' Estimated Employed', ' Estimated Labour Participation Rate (%)', ' Frequency', 'Year']].apply(pd.to_numeric)
y = df[' Estimated Unemployment Rate (%)'].apply(pd.to_numeric)

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.impute import SimpleImputer

# Impute missing values for numeric columns
imputer = SimpleImputer(strategy='mean')

# Applying imputation to X
X = imputer.fit_transform(df[[' Estimated Employed', ' Estimated Labour Participation Rate (%)', ' Frequency', 'Year']])
y = df[' Estimated Unemployment Rate (%)'].apply(pd.to_numeric, errors='coerce').fillna(df[' Estimated Unemployment Rate (%)'].mean())

# Verify no missing values remain
print(np.isnan(X).sum())
print(np.isnan(y).sum())

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
0
0
```

Linear Regression

```
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
lin_predictions = lin_reg.predict(X_test)

# Evaluate the model
lin_mse = mean_squared_error(y_test, lin_predictions)
lin_r2 = r2_score(y_test, lin_predictions)
print(f'Linear Regression Mean Squared Error: {lin_mse}')
print(f'Linear Regression R-squared: {lin_r2}')
```

```
Linear Regression Mean Squared Error: 88.39585039648902
Linear Regression R-squared: -0.010182415495096464
```

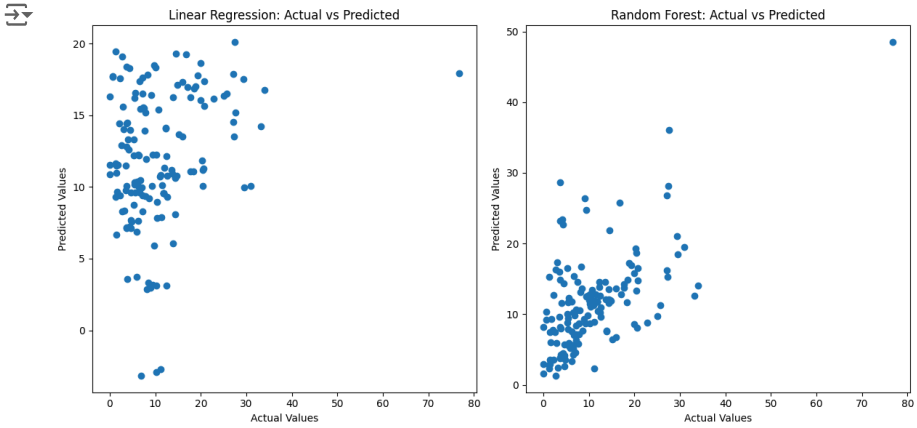
Random Forest Regressor

```
rf_reg = RandomForestRegressor(random_state=42)
rf_reg.fit(X_train, y_train)
rf_predictions = rf_reg.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_predictions)
rf_r2 = r2_score(y_test, rf_predictions)
print(f'Random Forest Mean Squared Error: {rf_mse}')
print(f'Random Forest R-squared: {rf_r2}')
```

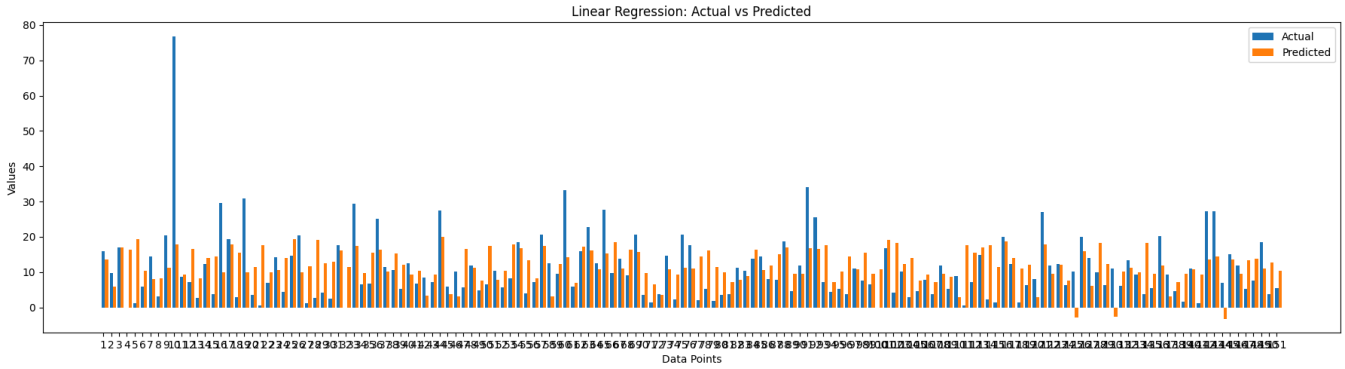
```
Random Forest Mean Squared Error: 56.49834796707557
Random Forest R-squared: 0.35434030709738196
```

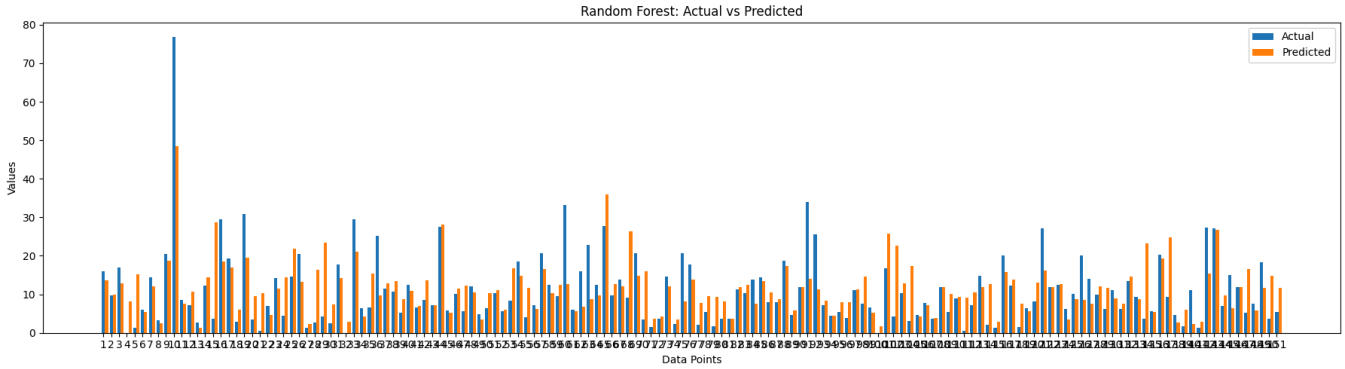
Ploting Linear Regression and Random Forest Regressor for understanding the model

```
import matplotlib.pyplot as plt

# Plot actual vs predicted values for each model
plt.figure(figsize=(18, 6))

# Linear Regression
plt.subplot(1, 3, 1)
plt.scatter(y_test, lin_predictions)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Linear Regression: Actual vs Predicted')

# Random Forest
plt.subplot(1, 3, 2)
plt.scatter(y_test, rf_predictions)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Random Forest: Actual vs Predicted')

plt.tight_layout()
plt.show()
```




```
import matplotlib.pyplot as plt
plt.figure(figsize=(18, 10))

# Linear Regression
plt.subplot(2, 1, 1)
plt.bar(np.arange(len(y_test)), y_test, width=0.4, label='Actual', align='center')
plt.bar(np.arange(len(lin_predictions)) + 0.4, lin_predictions, width=0.4, label='Predicted', align='center')
plt.xlabel('Data Points')
plt.ylabel('Values')
plt.title('Linear Regression: Actual vs Predicted')
plt.xticks(np.arange(len(y_test)), np.arange(1, len(y_test)+1))
plt.legend()

# Random Forest
plt.subplot(2, 1, 2)
plt.bar(np.arange(len(y_test)), y_test, width=0.4, label='Actual', align='center')
plt.bar(np.arange(len(rf_predictions)) + 0.4, rf_predictions, width=0.4, label='Predicted', align='center')
plt.xlabel('Data Points')
plt.ylabel('Values')
plt.title('Random Forest: Actual vs Predicted')
plt.xticks(np.arange(len(y_test)), np.arange(1, len(y_test)+1))
plt.legend()

plt.tight_layout()
plt.show()
```


Linear Regression: Actual vs Predicted
Actual
Predicted
Values
Data Points


Random Forest: Actual vs Predicted
Actual
Predicted
Values
Data Points