

Load Data and import Dependencies

```
In [2]: !pip install ucimlrepo
!pip install xlrd
```

```
Requirement already satisfied: ucimlrepo in c:\users\administrator\anaconda3\lib\site-packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in c:\users\administrator\anaconda3\lib\site-packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in c:\users\administrator\anaconda3\lib\site-packages (from ucimlrepo) (2025.1.31)
Requirement already satisfied: numpy>=1.26.0 in c:\users\administrator\anaconda3\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\administrator\anaconda3\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\administrator\anaconda3\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\administrator\anaconda3\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\administrator\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrepo) (1.16.0)
Requirement already satisfied: xlrd in c:\users\administrator\anaconda3\lib\site-packages (2.0.1)
```

```
In [3]: from ucimlrepo import fetch_ucirepo
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, Lasso, LinearRegression
```

```
In [4]: # fetch dataset from uci
concrete_compressive_strength = fetch_ucirepo(id=165)

# data (as pandas dataframes)
X = concrete_compressive_strength.data.features
y = concrete_compressive_strength.data.targets

# metadata
print(concrete_compressive_strength.metadata)

# variable information
print(concrete_compressive_strength.variables)
```

```
{'uci_id': 165, 'name': 'Concrete Compressive Strength', 'repository_url': 'https://
archive.ics.uci.edu/dataset/165/concrete+compressive+strength', 'data_url': 'http
s://archive.ics.uci.edu/static/public/165/data.csv', 'abstract': 'Concrete is the mo
st important material in civil engineering. The concrete compressive strength is a h
ighly nonlinear function of age and ingredients. ', 'area': 'Physics and Chemistry',
'tasks': ['Regression'], 'characteristics': ['Multivariate'], 'num_instances': 1030,
'num_features': 8, 'feature_types': ['Real'], 'demographics': [], 'target_col': ['Co
ncrete compressive strength'], 'index_col': None, 'has_missing_values': 'no', 'missi
ng_values_symbol': None, 'year_of_dataset_creation': 1998, 'last_updated': 'Sun Feb
11 2024', 'dataset_doi': '10.24432/C5PK67', 'creators': ['I-Cheng Yeh'], 'intro_pape
r': {'ID': 383, 'type': 'NATIVE', 'title': 'Modeling of strength of high-performance
concrete using artificial neural networks', 'authors': 'I. Yeh', 'venue': 'Cement an
d Concrete Research, Vol. 28, No. 12', 'year': 1998, 'journal': None, 'DOI': '10.101
6/S0008-8846(98)00165-3', 'URL': 'https://www.semanticscholar.org/paper/9310cae70452
ea11465f338483e79cc36a68881c', 'sha': None, 'corpus': None, 'arxiv': None, 'mag': No
ne, 'acl': None, 'pmid': None, 'pmcid': None}, 'additional_info': {'summary': 'Numbe
r of instances \t1030\r\nNumber of Attributes\t9\r\nAttribute breakdown\t8 quantitat
ive input variables, and 1 quantitative output variable\r\nMissing Attribute Values
\tNone \r\n', 'purpose': None, 'funded_by': None, 'instances_represent': None, 'reco
mmended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': Non
e, 'variable_info': 'Given are the variable name, variable type, the measurement uni
t and a brief description. The concrete compressive strength is the regression probl
em. The order of this listing corresponds to the order of numerals along the rows of
the database. \r\n\r\nName -- Data Type -- Measurement -- Description\r\n\r\nCement
(component 1) -- quantitative -- kg in a m3 mixture -- Input Variable\r\nBlast Furna
ce Slag (component 2) -- quantitative -- kg in a m3 mixture -- Input Variable\r\nFly
Ash (component 3) -- quantitative -- kg in a m3 mixture -- Input Variable\r\nWater
(component 4) -- quantitative -- kg in a m3 mixture -- Input Variable\r\nSuperplast
icizer (component 5) -- quantitative -- kg in a m3 mixture -- Input Variable\r\nCoar
se Aggregate (component 6) -- quantitative -- kg in a m3 mixture -- Input Variable
\r\nFine Aggregate (component 7)\t -- quantitative -- kg in a m3 mixture -- Input V
ariable\r\nAge -- quantitative -- Day (1~365) -- Input Variable\r\nConcrete compres
sive strength -- quantitative -- MPa -- Output Variable\r\n\r\n', 'citation': None}}
```

	name	role	type	demographic	description \
0	Cement	Feature	Continuous	None	None
1	Blast Furnace Slag	Feature	Integer	None	None
2	Fly Ash	Feature	Continuous	None	None
3	Water	Feature	Continuous	None	None
4	Superplasticizer	Feature	Continuous	None	None
5	Coarse Aggregate	Feature	Continuous	None	None
6	Fine Aggregate	Feature	Continuous	None	None
7	Age	Feature	Integer	None	None
8	Concrete compressive strength	Target	Continuous	None	None

	units	missing_values
0	kg/m^3	no
1	kg/m^3	no
2	kg/m^3	no
3	kg/m^3	no
4	kg/m^3	no
5	kg/m^3	no
6	kg/m^3	no
7	day	no
8	MPa	no

In [5]: X

Out[5]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360
...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28

1030 rows × 8 columns

In [6]: y

Out[6]:

Concrete compressive strength	
0	79.99
1	61.89
2	40.27
3	41.05
4	44.30
...	...
1025	44.28
1026	31.18
1027	23.70
1028	32.77
1029	32.40

1030 rows × 1 columns

```
In [7]: # Import data from local source
# file = r'data/Concrete_Data.xls'
```

```
df = pd.read_excel('data/Concrete_Data.xls')
df
```

Out[7]:

	Cement (component 1)(kg in a m³ mixture)	Blast Furnace Slag (component 2)(kg in a m³ mixture)	Fly Ash (component 3)(kg in a m³ mixture)	Water (component 4)(kg in a m³ mixture)	Superplasticizer (component 5) (kg in a m³ mixture)	Coarse Aggregate (component 6)(kg in a m³ mixture)
0	540.0	0.0	0.0	162.0	2.5	1040.0
1	540.0	0.0	0.0	162.0	2.5	1055.0
2	332.5	142.5	0.0	228.0	0.0	932.0
3	332.5	142.5	0.0	228.0	0.0	932.0
4	198.6	132.4	0.0	192.0	0.0	978.4
...
1025	276.4	116.0	90.3	179.6	8.9	870.1
1026	322.2	0.0	115.6	196.0	10.4	817.9
1027	148.5	139.4	108.6	192.7	6.1	892.4
1028	159.1	186.7	0.0	175.6	11.3	989.6
1029	260.9	100.5	78.3	200.6	8.6	864.5

1030 rows × 9 columns

```
In [8]: df.columns = df.columns.str.strip()
```

```
In [9]: # Renaming columns with direct mapping
df.rename(columns={
    'Cement (component 1)(kg in a m^3 mixture)': 'Cement',
    'Blast Furnace Slag (component 2)(kg in a m^3 mixture)': 'Blast Furnace Slag',
    'Fly Ash (component 3)(kg in a m^3 mixture)': 'Fly Ash',
    'Water (component 4)(kg in a m^3 mixture)': 'Water',
    'Superplasticizer (component 5)(kg in a m^3 mixture)': 'Superplasticizer',
    'Coarse Aggregate (component 6)(kg in a m^3 mixture)': 'Coarse Aggregate',
    'Fine Aggregate (component 7)(kg in a m^3 mixture)': 'Fine Aggregate',
    'Age(day)': 'Age',
    'Concrete compressive strength(MPa, megapascals)': 'Concrete compressive streng
}, inplace=True)
```

```
In [10]: # Renaming columns with regex
import re

rename_dict = {
    r'.*Cement.*': 'Cement',
    r'.*Blast Furnace Slag.*': 'Blast Furnace Slag',
    r'.*Fly Ash.*': 'Fly Ash',
```

```

r'.*Water.*': 'Water',
r'.*Superplasticizer.*': 'Superplasticizer',
r'.*Coarse Aggregate.*': 'Coarse Aggregate',
r'.*Fine Aggregate.*': 'Fine Aggregate',
r'.*Age.*': 'Age',
r'.*Concrete compressive strength.*': 'Concrete compressive strength'
}

df.columns = [next((v for k, v in rename_dict.items() if re.match(k, col)), col) for col in df.columns]

```

In [11]: df

Out[11]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Concrete compressive strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.1
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.1
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.1
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.1
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.1
...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.1
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.1
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.1
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.1
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.1

1030 rows × 9 columns

Data Analysis and Visualization

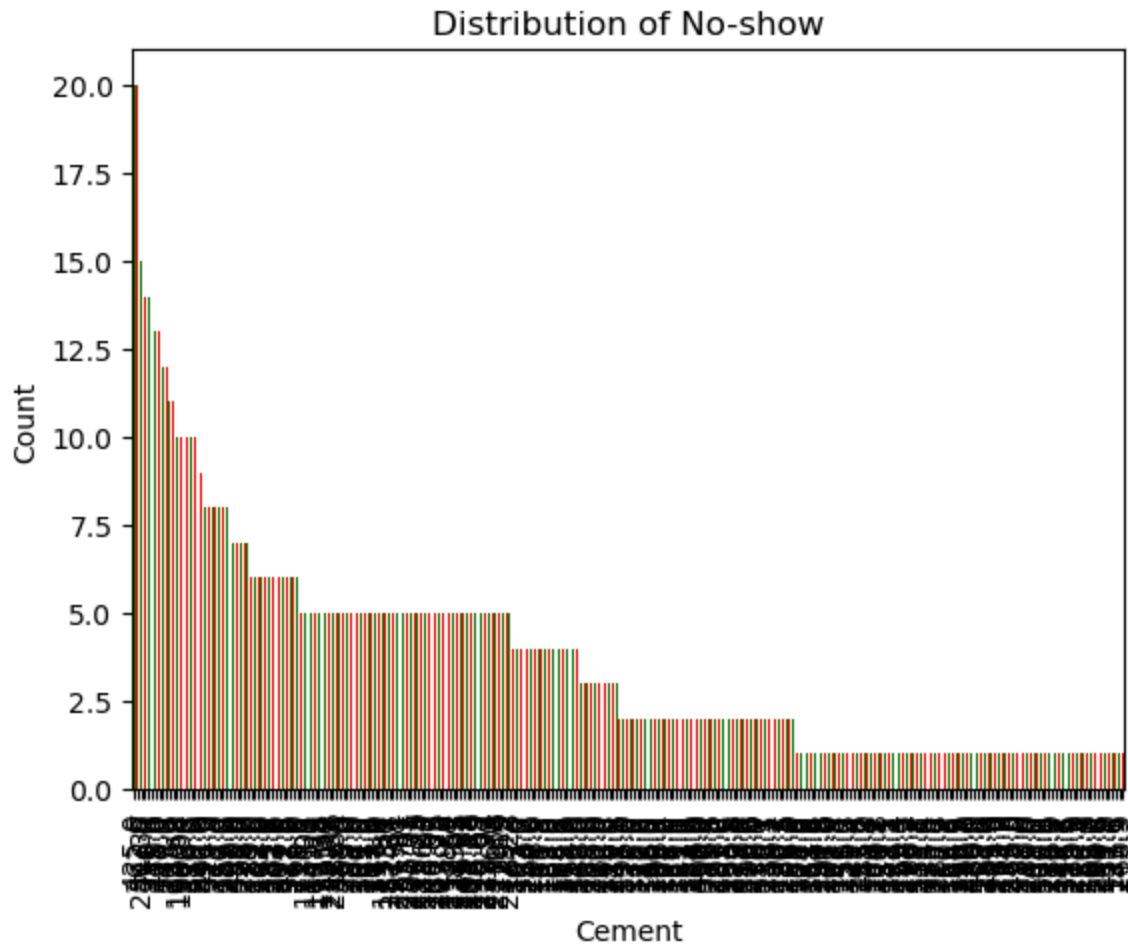
In [13]: df.isnull()

Out[13]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Compressive Strength
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
...
1025	False	False	False	False	False	False	False	False	
1026	False	False	False	False	False	False	False	False	
1027	False	False	False	False	False	False	False	False	
1028	False	False	False	False	False	False	False	False	
1029	False	False	False	False	False	False	False	False	

1030 rows × 9 columns

```
In [14]: # Plotting the distribution of No-show
df['Cement'].value_counts().plot(kind='bar', color=['green', 'red'])
plt.title('Distribution of No-show')
plt.xlabel('Cement')
plt.ylabel('Count')
plt.show()
```



```
In [15]: # Create a figure with 3x3 subplots, setting the overall figure size to 18x15 inches
fig, axes = plt.subplots(3, 3, figsize=(18, 15))

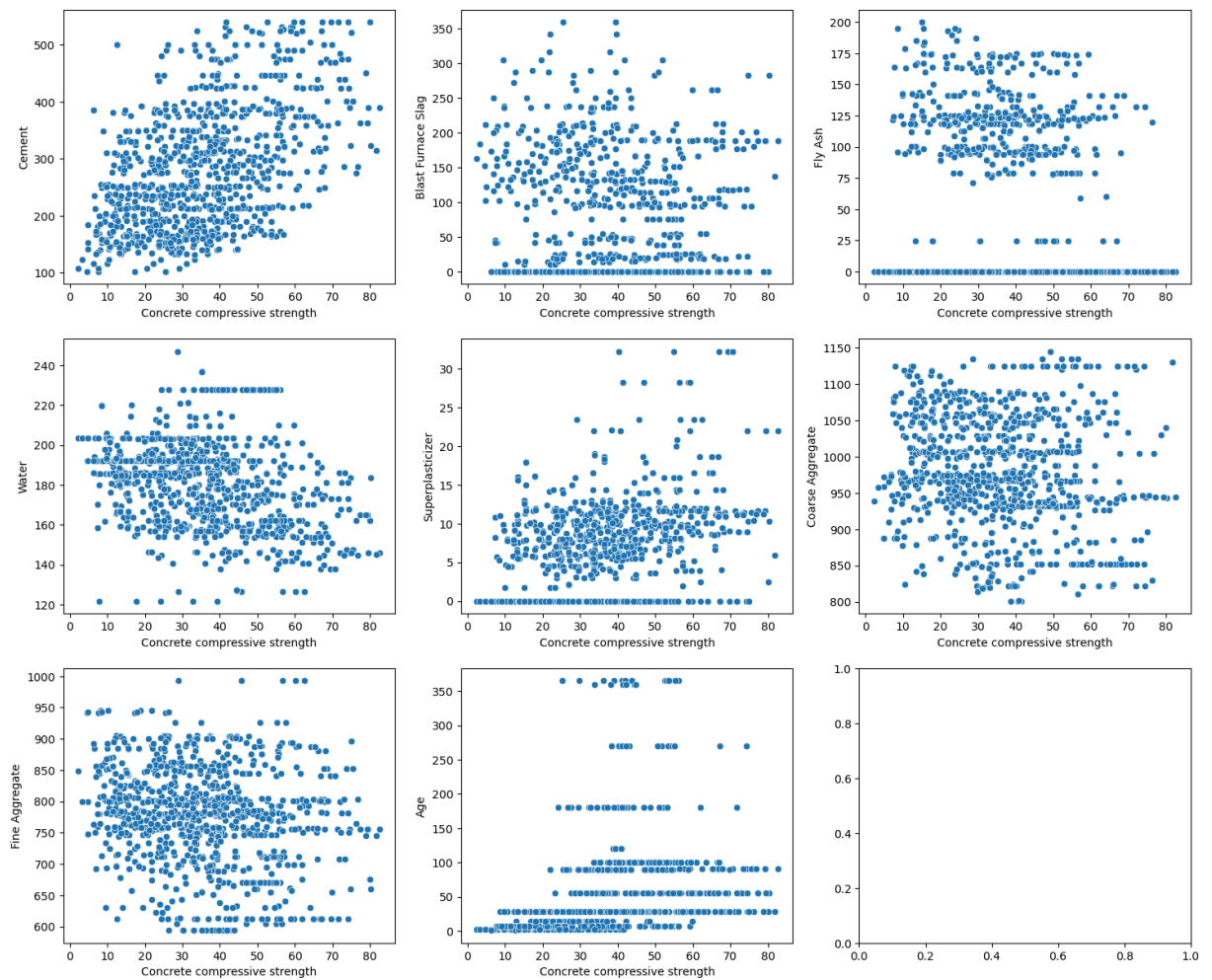
# Set the main title for the entire figure
fig.suptitle('Understanding Concrete Compressive Strength - 3 x 3 axes Box plot')

# Create scatter plots for each feature vs Concrete compressive strength
# Row 0: Cement, Blast Furnace Slag, Fly Ash
sns.scatterplot(ax=axes[0, 0], data=df, x=df['Concrete compressive strength'], y=df['Cement'])
sns.scatterplot(ax=axes[0, 1], data=df, x=df['Concrete compressive strength'], y=df['Blast Furnace Slag'])
sns.scatterplot(ax=axes[0, 2], data=df, x=df['Concrete compressive strength'], y=df['Fly Ash'])

# Row 1: Water, Superplasticizer, Coarse Aggregate
sns.scatterplot(ax=axes[1, 0], data=df, x=df['Concrete compressive strength'], y=df['Water'])
sns.scatterplot(ax=axes[1, 1], data=df, x=df['Concrete compressive strength'], y=df['Superplasticizer'])
sns.scatterplot(ax=axes[1, 2], data=df, x=df['Concrete compressive strength'], y=df['Coarse Aggregate'])

# Row 2: Fine Aggregate, Age
sns.scatterplot(ax=axes[2, 0], data=df, x=df['Concrete compressive strength'], y=df['Fine Aggregate'])
sns.scatterplot(ax=axes[2, 1], data=df, x=df['Concrete compressive strength'], y=df['Age'])
```

```
Out[15]: <Axes: xlabel='Concrete compressive strength', ylabel='Age'>
```



```
In [16]: features = df.drop('Concrete compressive strength', axis=1)
```

```
In [17]: # generating pairwise correlation
corr = features.corr()

# Displaying dataframe as an heatmap
# with diverging colourmap as coolwarm
corr.style.background_gradient(cmap = 'coolwarm')
```


Out[17]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Age
Cement	1.000000	-0.275193	-0.397475	-0.081544	0.092771	-0.109356	-
Blast Furnace Slag	-0.275193	1.000000	-0.323569	0.107286	0.043376	-0.283998	-
Fly Ash	-0.397475	-0.323569	1.000000	-0.257044	0.377340	-0.009977	-
Water	-0.081544	0.107286	-0.257044	1.000000	-0.657464	-0.182312	-
Superplasticizer	0.092771	0.043376	0.377340	-0.657464	1.000000	-0.266303	-
Coarse Aggregate	-0.109356	-0.283998	-0.009977	-0.182312	-0.266303	1.000000	-
Fine Aggregate	-0.222720	-0.281593	0.079076	-0.450635	0.222501	-0.178506	-
Age	0.081947	-0.044246	-0.154370	0.277604	-0.192717	-0.003016	-

Split your dataset

Feature Engineering and Cleaning

In [20]: `df.dropna()`

Out[20]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Compressive Strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.0
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.0
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.0
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.0
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.0
...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.0
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.0
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.0
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.0
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.0

1030 rows × 9 columns

In [21]: `df.shape`

Out[21]: (1030, 9)

In [22]: `X = features = df.drop('Concrete compressive strength', axis=1)`
`y = target = df['Concrete compressive strength']`

80 - 20 split

In [24]: `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)`

Model Selection & Engineering

In [26]: `linear = LinearRegression()`
`linear.fit(X_train, y_train)`

`print("Accuracy score on training {:.4f}".format(linear.score(X_train, y_train)))`
`print("Accuracy score on testing {:.4f}".format(linear.score(X_test, y_test)))`

Accuracy score on training 0.6105

Accuracy score on testing 0.6275

In [27]: `# Model Selection`
`ridge = Ridge(max_iter=1000000)`
`ridge.fit(X_train, y_train)`

Out[27]:

```
▼ Ridge ⓘ ?  
Ridge(max_iter=1000000)
```

```
In [28]: ridge.fit(X_train, y_train)  
print("Accuracy score on training {:.4f}".format(ridge.score(X_train,y_train)))  
print("Accuracy score on testing {:.4f}".format(ridge.score(X_test,y_test)))
```

Accuracy score on training 0.6105
Accuracy score on testing 0.6275

```
In [29]: # Model Selection - best fit for ridge  
ridge100 = Ridge(alpha=100, max_iter=1000000)  
ridge100.fit(X_train, y_train)  
  
ridge.fit(X_train, y_train)  
print("Accuracy score on training {:.4f}".format(ridge100.score(X_train,y_train)))  
print("Accuracy score on testing {:.4f}".format(ridge100.score(X_test,y_test)))
```

Accuracy score on training 0.6105
Accuracy score on testing 0.6276

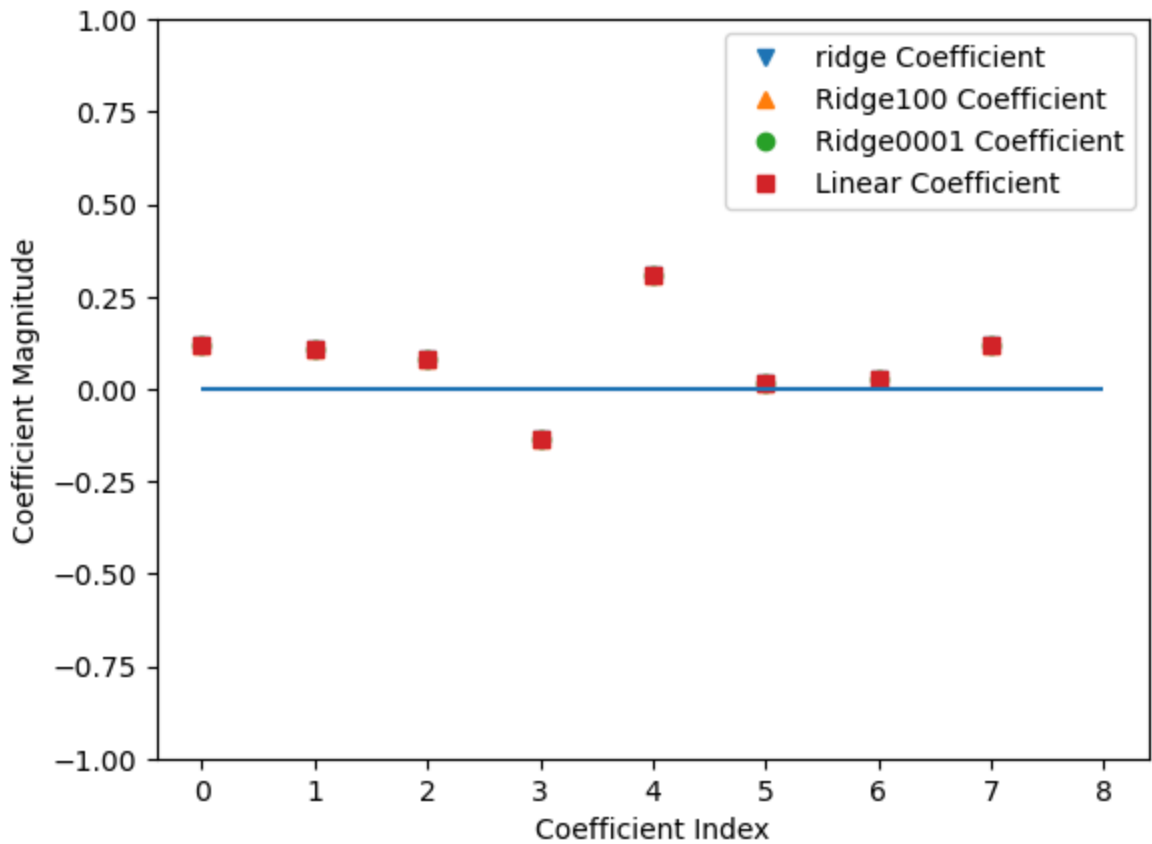
```
In [30]: # Model Selection  
ridge0001 = Ridge(alpha=0.0001, max_iter=1000000)  
ridge0001.fit(X_train, y_train)  
  
ridge.fit(X_train, y_train)  
print("Accuracy score on training {:.4f}".format(ridge0001.score(X_train,y_train)))  
print("Accuracy score on testing {:.4f}".format(ridge0001.score(X_test,y_test)))
```

Accuracy score on training 0.6105
Accuracy score on testing 0.6275

Ridge Regularization Impact vs LinearRegression

```
In [32]: plt.plot(ridge.coef_, 'v', label="ridge Coefficient")  
plt.plot(ridge100.coef_, '^', label="Ridge100 Coefficient")  
plt.plot(ridge0001.coef_, 'o', label="Ridge0001 Coefficient")  
  
plt.plot(linear.coef_, 's', label="Linear Coefficient")  
plt.hlines(0,0, len(linear.coef_))  
plt.ylabel("Coefficient Magnitude")  
plt.xlabel("Coefficient Index")  
plt.ylim(-1,1)  
plt.legend()
```

Out[32]: <matplotlib.legend.Legend at 0x27950f9d550>



```
In [33]: from sklearn.linear_model import Lasso
lasso = Lasso(max_iter=1000000)

lasso.fit(X_train, y_train)
print("Accuracy score on training {:.4f}".format(lasso.score(X_train,y_train)))
print("Accuracy score on testing {:.4f}".format(lasso.score(X_test,y_test)))
print("Number of features {}".format(np.sum(lasso.coef_ != 0)))
```

Accuracy score on training 0.6102
Accuracy score on testing 0.6276
Number of features 8

```
In [34]: # Lasso0001 - Best fit for lasso
lasso0001 = Lasso(alpha=0.0001, max_iter=1000000)
lasso0001.fit(X_train, y_train)
print("Accuracy score on training {:.4f}".format(lasso0001.score(X_train,y_train)))
print("Accuracy score on testing {:.4f}".format(lasso0001.score(X_test,y_test)))
print("Number of features {}".format(np.sum(lasso0001.coef_ != 0)))
```

Accuracy score on training 0.6105
Accuracy score on testing 0.6275
Number of features 8

```
In [35]: lasso10 = Lasso(alpha=10, max_iter=1000000)

lasso10.fit(X_train, y_train)
print("Accuracy score on training {:.4f}".format(lasso10.score(X_train,y_train)))
print("Accuracy score on testing {:.4f}".format(lasso10.score(X_test,y_test)))
print("Number of features {}".format(np.sum(lasso10.coef_ != 0)))
```

Accuracy score on training 0.6046
Accuracy score on testing 0.6214
Number of features 6

```
In [36]: # Lasso100 - Lowest
lasso100 = Lasso(alpha=100,max_iter=1000000)

lasso100.fit(X_train, y_train)
print("Accuracy score on training {:.4f}".format(lasso100.score(X_train,y_train)))
print("Accuracy score on testing {:.4f}".format(lasso100.score(X_test,y_test)))
print("Number of features {}".format(np.sum(lasso100.coef_ != 0)))
```

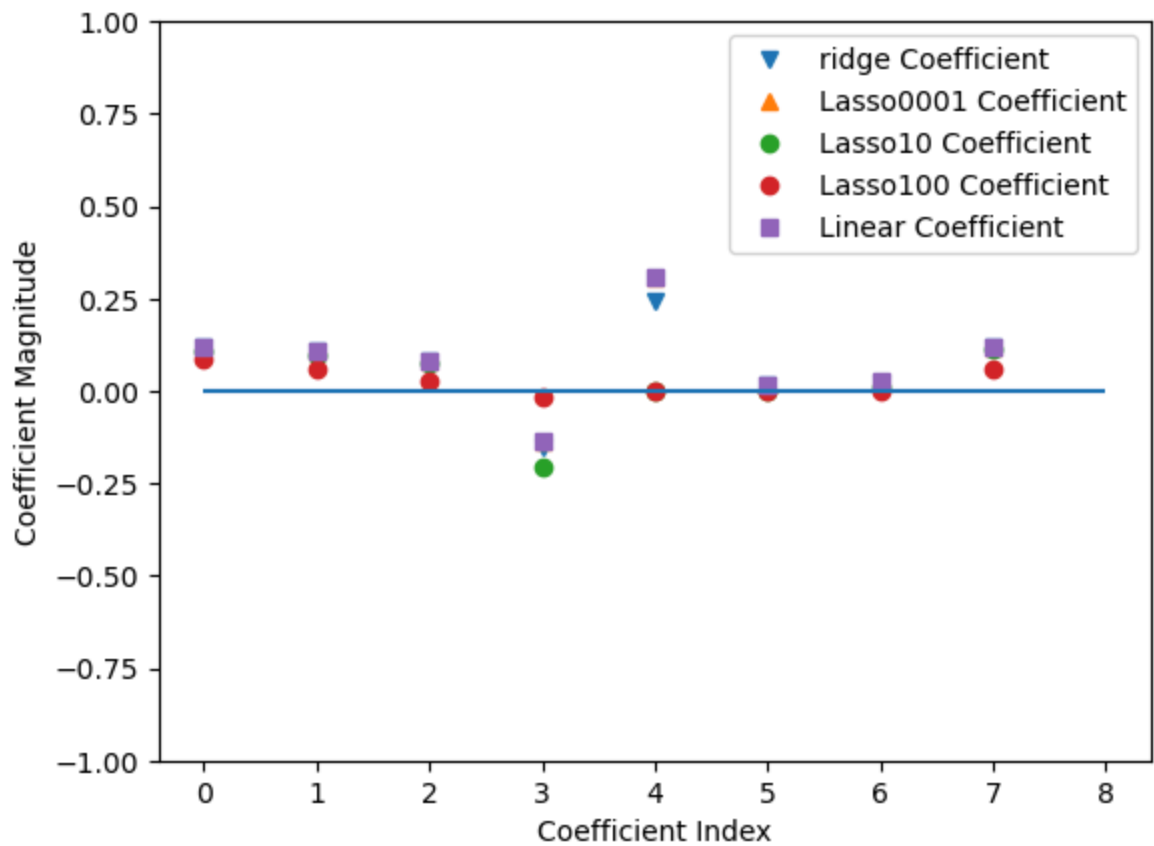
Accuracy score on training 0.4653
Accuracy score on testing 0.4398
Number of features 5

Lasso Regularization Impact vs LinearRegression

```
In [38]: plt.plot(lasso.coef_, 'v', label="ridge Coefficient")
plt.plot(lasso0001.coef_, '^', label="Lasso0001 Coefficient")
plt.plot(lasso10.coef_, 'o', label="Lasso10 Coefficient")
plt.plot(lasso100.coef_, 'o', label="Lasso100 Coefficient")

plt.plot(linear.coef_, 's', label="Linear Coefficient")
plt.hlines(0,0, len(linear.coef_))
plt.ylabel("Coefficient Magnitude")
plt.xlabel("Coefficient Index")
plt.ylim(-1,1)
plt.legend()
```

Out[38]: <matplotlib.legend.Legend at 0x27951029cd0>



Predictions using best fit model

```
In [40]: y_pred = linear.predict(X_test)
         results_linear = pd.Series(y_pred)
         results_linear
```

```
Out[40]: 0      59.657163
         1      52.037144
         2      63.519839
         3      51.571366
         4      17.220160
         ...
         201    56.000405
         202    17.486689
         203    49.087594
         204    54.199513
         205    31.467891
         Length: 206, dtype: float64
```

```
In [41]: y_pred_ridge = ridge100.predict(X_test)
         results_ridge = pd.Series(y_pred_ridge)
         results_ridge
```

```
Out[41]: 0      59.653095
         1      52.032451
         2      63.497376
         3      51.566683
         4      17.219626
         ...
        201     56.000707
        202     17.495513
        203     49.087525
        204     54.217546
        205     31.473652
        Length: 206, dtype: float64
```

```
In [42]: y_pred_lasso = lasso0001.predict(X_test)
         results_lasso = pd.Series(y_pred_lasso)
         results_lasso
```

```
Out[42]: 0      59.657153
         1      52.037126
         2      63.519781
         3      51.571348
         4      17.220164
         ...
        201     56.000404
        202     17.486719
        203     49.087587
        204     54.199560
        205     31.467914
        Length: 206, dtype: float64
```

```
In [43]: import pandas as pd
         import matplotlib.pyplot as plt

         # Assuming df_graphs is already defined as:
         df_graphs = pd.DataFrame({'Linear_pred': results_linear, 'Lasso_pred': results_lasso})

         # Assuming y_true contains the actual values
         # Add the actual values to the DataFrame
         df_graphs['Actual'] = y_test

         # Plot each prediction vs actual values
         plt.figure(figsize=(14, 8))

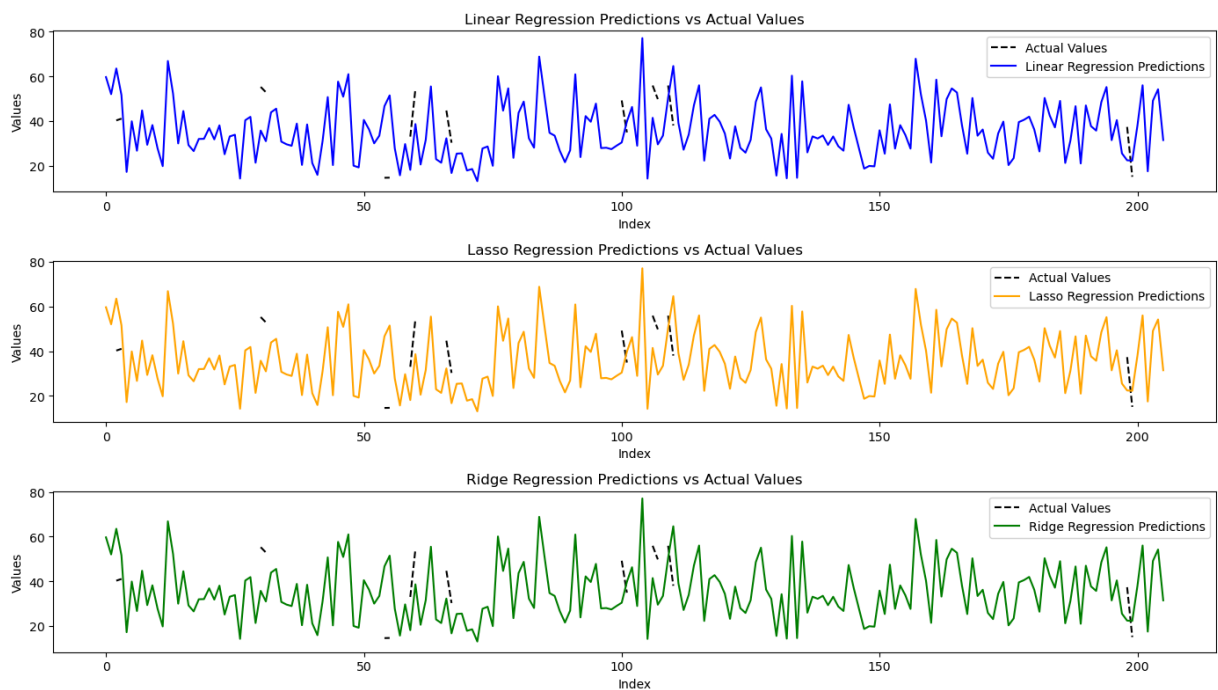
         # Plot Linear Regression predictions vs actual values
         plt.subplot(3, 1, 1)
         plt.plot(df_graphs['Actual'], label='Actual Values', color='black', linestyle='--')
         plt.plot(df_graphs['Linear_pred'], label='Linear Regression Predictions', color='blue')
         plt.title('Linear Regression Predictions vs Actual Values')
         plt.xlabel('Index')
         plt.ylabel('Values')
         plt.legend()

         # Plot Lasso Regression predictions vs actual values
         plt.subplot(3, 1, 2)
         plt.plot(df_graphs['Actual'], label='Actual Values', color='black', linestyle='--')
         plt.plot(df_graphs['Lasso_pred'], label='Lasso Regression Predictions', color='orange')
```

```
plt.title('Lasso Regression Predictions vs Actual Values')
plt.xlabel('Index')
plt.ylabel('Values')
plt.legend()

# Plot Ridge Regression predictions vs actual values
plt.subplot(3, 1, 3)
plt.plot(df_graphs['Actual'], label='Actual Values', color='black', linestyle='--')
plt.plot(df_graphs['Ridge_pred'], label='Ridge Regression Predictions', color='green')
plt.title('Ridge Regression Predictions vs Actual Values')
plt.xlabel('Index')
plt.ylabel('Values')
plt.legend()

# Adjust layout for better spacing
plt.tight_layout()
plt.show()
```



In [44]: `!pip install nbconvert[webpdf]`

Requirement already satisfied: nbconvert[webpdf] in c:\users\administrator\anaconda3\lib\site-packages (7.16.4)

Requirement already satisfied: beautifulsoup4 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (4.12.3)

Requirement already satisfied: bleach!=5.0.0 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (4.1.0)

Requirement already satisfied: defusedxml in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.7.1)

Requirement already satisfied: Jinja2>=3.0 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (3.1.4)

Requirement already satisfied: jupyter-core>=4.7 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (5.7.2)

Requirement already satisfied: jupyterlab-pygments in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.1.2)

Requirement already satisfied: MarkupSafe>=2.0 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (2.1.3)

Requirement already satisfied: mistune<4,>=2.0.3 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (2.0.4)

Requirement already satisfied: nbclient>=0.5.0 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.8.0)

Requirement already satisfied: nbformat>=5.7 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (5.10.4)

Requirement already satisfied: packaging in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (24.1)

Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (1.5.0)

Requirement already satisfied: pygments>=2.4.1 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (2.15.1)

Requirement already satisfied: tinycss2 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (1.2.1)

Requirement already satisfied: traitlets>=5.1 in c:\users\administrator\anaconda3\lib\site-packages (from nbconvert[webpdf]) (5.14.3)

Collecting playwright (from nbconvert[webpdf])

Using cached playwright-1.50.0-py3-none-win_amd64.whl.metadata (3.5 kB)

Requirement already satisfied: six>=1.9.0 in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (1.16.0)

Requirement already satisfied: webencodings in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (0.5.1)

Requirement already satisfied: platformdirs>=2.5 in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (3.10.0)

Requirement already satisfied: pywin32>=300 in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (305.1)

Requirement already satisfied: jupyter-client>=6.1.12 in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (8.6.0)

Requirement already satisfied: fastjsonschema>=2.15 in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (2.16.2)

Requirement already satisfied: jsonschema>=2.6 in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (4.23.0)

Requirement already satisfied: soupsieve>1.2 in c:\users\administrator\anaconda3\lib\site-packages (from playwright) (2.5)

Collecting pyee<13,>=12 (from playwright)

Using cached pyee-12.1.1-py3-none-any.whl.metadata (2.9 kB)

Collecting greenlet<4.0.0,>=3.1.1 (from playwright)

Using cached greenlet-3.1.1-cp312-cp312-win_amd64.whl.metadata (3.9 kB)

Requirement already satisfied: attrs>=22.2.0 in c:\users\administrator\anaconda3\lib\site-packages (from greenlet) (23.1.0)

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

	28.0/34.8 MB	71.0 kB/s	eta 0:01:35
	28.0/34.8 MB	71.0 kB/s	eta 0:01:35
	28.0/34.8 MB	71.0 kB/s	eta 0:01:35
	28.3/34.8 MB	70.6 kB/s	eta 0:01:32
	28.3/34.8 MB	70.6 kB/s	eta 0:01:32
	28.3/34.8 MB	70.6 kB/s	eta 0:01:32
	28.6/34.8 MB	75.3 kB/s	eta 0:01:23
	28.6/34.8 MB	75.3 kB/s	eta 0:01:23
	28.6/34.8 MB	75.3 kB/s	eta 0:01:23
	28.6/34.8 MB	75.3 kB/s	eta 0:01:23
	28.8/34.8 MB	82.5 kB/s	eta 0:01:13
	28.8/34.8 MB	82.5 kB/s	eta 0:01:13
	28.8/34.8 MB	82.5 kB/s	eta 0:01:13
	28.8/34.8 MB	82.5 kB/s	eta 0:01:13
	28.8/34.8 MB	82.5 kB/s	eta 0:01:13
	28.8/34.8 MB	82.5 kB/s	eta 0:01:13
	28.8/34.8 MB	82.5 kB/s	eta 0:01:13
	29.1/34.8 MB	82.6 kB/s	eta 0:01:09
	29.1/34.8 MB	82.6 kB/s	eta 0:01:09
	29.1/34.8 MB	82.6 kB/s	eta 0:01:09
	29.1/34.8 MB	82.6 kB/s	eta 0:01:09
	29.1/34.8 MB	82.6 kB/s	eta 0:01:09
	29.4/34.8 MB	88.3 kB/s	eta 0:01:02
	29.4/34.8 MB	88.3 kB/s	eta 0:01:02
	29.4/34.8 MB	88.3 kB/s	eta 0:01:02
	29.4/34.8 MB	88.3 kB/s	eta 0:01:02
	29.4/34.8 MB	88.3 kB/s	eta 0:01:02
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.6/34.8 MB	110.7 kB/s	eta 0:00:47
	29.9/34.8 MB	114.3 kB/s	eta 0:00:43
	29.9/34.8 MB	114.3 kB/s	eta 0:00:43
	29.9/34.8 MB	114.3 kB/s	eta 0:00:43
	29.9/34.8 MB	114.3 kB/s	eta 0:00:43
	30.1/34.8 MB	119.4 kB/s	eta 0:00:39
	30.1/34.8 MB	119.4 kB/s	eta 0:00:39
	30.1/34.8 MB	119.4 kB/s	eta 0:00:39
	30.1/34.8 MB	119.4 kB/s	eta 0:00:39
	30.1/34.8 MB	119.4 kB/s	eta 0:00:39
	30.4/34.8 MB	124.8 kB/s	eta 0:00:36
	30.4/34.8 MB	124.8 kB/s	eta 0:00:36
	30.4/34.8 MB	124.8 kB/s	eta 0:00:36
	30.4/34.8 MB	124.8 kB/s	eta 0:00:36
	30.7/34.8 MB	130.2 kB/s	eta 0:00:32
	30.7/34.8 MB	130.2 kB/s	eta 0:00:32
	30.7/34.8 MB	130.2 kB/s	eta 0:00:32
	30.9/34.8 MB	136.6 kB/s	eta 0:00:29
	30.9/34.8 MB	136.6 kB/s	eta 0:00:29
	30.9/34.8 MB	136.6 kB/s	eta 0:00:29

[illegible]

[illegible]

```
----- - 33.6/34.8 MB 140.5 kB/s eta 0:00:09
----- - 33.8/34.8 MB 141.4 kB/s eta 0:00:07
----- - 33.8/34.8 MB 141.4 kB/s eta 0:00:07
----- - 33.8/34.8 MB 141.4 kB/s eta 0:00:07
----- 34.1/34.8 MB 146.1 kB/s eta 0:00:05
----- 34.1/34.8 MB 146.1 kB/s eta 0:00:05
----- 34.1/34.8 MB 146.1 kB/s eta 0:00:05
----- 34.3/34.8 MB 151.8 kB/s eta 0:00:03
----- 34.3/34.8 MB 151.8 kB/s eta 0:00:03
----- 34.3/34.8 MB 151.8 kB/s eta 0:00:03
----- 34.3/34.8 MB 151.8 kB/s eta 0:00:03
----- 34.6/34.8 MB 153.1 kB/s eta 0:00:02
----- 34.6/34.8 MB 153.1 kB/s eta 0:00:02
----- 34.6/34.8 MB 153.1 kB/s eta 0:00:02
----- 34.6/34.8 MB 153.1 kB/s eta 0:00:02
----- 34.6/34.8 MB 153.1 kB/s eta 0:00:02
----- 34.6/34.8 MB 153.1 kB/s eta 0:00:02
----- 34.8/34.8 MB 149.0 kB/s eta 0:00:00
```

Using cached greenlet-3.1.1-cp312-cp312-win_amd64.whl (299 kB)

Downloading pyee-12.1.1-py3-none-any.whl (15 kB)

Installing collected packages: pyee, greenlet, playwright

Attempting uninstall: greenlet

Found existing installation: greenlet 3.0.1

Uninstalling greenlet-3.0.1:

Successfully uninstalled greenlet-3.0.1

Successfully installed greenlet-3.1.1 playwright-1.50.0 pyee-12.1.1

WARNING: Failed to remove contents in a temporary directory 'C:\Users\Administrato
r\Anaconda3\Lib\site-packages\~reenlet'.
You can safely remove it manually.

In []: !jupyter nbconvert --to webpdf --allow-chromium-download Concrete_Comprehensive.ipynb

In []: