```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error as mae

import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('/content/train.csv.zip')
display(df.head())
display(df.tail())
df.shape
#column of the dataset contains which type of data.
df.info()
df.describe()


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error as mae

import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('/content/train.csv.zip')
display(df.head())
```

```
display(df.tail())
df.shape
df.info()
df.describe()
parts = df["date"].str.split("-", n = 3, expand = True)
df["year"]= parts[0].astype('int')
df["month"]= parts[1].astype('int')
df["day"]= parts[2].astype('int')
df.head()
from datetime import datetime
import calendar
def weekend_or_weekday(year,month,day):

  d = datetime(year,month,day)
  if d.weekday()>4:
    return 1
  else:
    return 0

df['weekend'] = df.apply(lambda x:weekend_or_weekday(x['year'], x['month'],
x['day']), axis=1)
df.head()


plt.subplots(figsize=(12, 5))
plt.subplot(1, 2, 1)
sb.distplot(df['sales'])

plt.subplot(1, 2, 2)
sb.boxplot(df['sales'])
plt.show()
plt.figure(figsize=(10, 10))
sb.heatmap(df.corr() > 0.8,
    annot=True,
    cbar=False)
plt.show()
features = df.drop(['sales', 'year'], axis=1)
target = df['sales'].values


X_train, X_val, Y_train, Y_val = train_test_split(features, target,
                      test_size = 0.05,
                      random_state=22)
```

```
X_train.shape, X_val.shape

#for weekday
from datetime import date
import holidays
import pandas as pd

# Sample DataFrame with 'date', 'store', 'year', 'month', 'id', and 'item' columns
df = pd.DataFrame({
    'date': ['2024-01-01', '2024-02-15', '2024-04-14', '2024-04-15'],
    'store': [1, 2, 1, 2],
    'year': [2024, 2024, 2024, 2024],
    'month': [1, 2, 4, 4],
    'id': [101, 102, 103, 104],
    'item': ['A', 'B', 'A', 'B']
})

def is_holiday(x):
    india_holidays = holidays.CountryHoliday('IN')
    return 1 if x in india_holidays else 0

df['date'] = pd.to_datetime(df['date'])
df['day_of_week'] = df['date'].dt.day_name()
df['is_weekend'] = df['date'].dt.dayofweek // 5  # 1 for Saturday or Sunday, 0 for
weekdays
df['is_holiday'] = df['date'].apply(is_holiday)

print(df.head())
df['m1'] = np.sin(df['month'] * (2 * np.pi / 12))
df['m2'] = np.cos(df['month'] * (2 * np.pi / 12))
df.head()
from datetime import datetime
import holidays
import pandas as pd

# Assuming df is your DataFrame with 'year', 'month', 'day', and other columns
df = pd.DataFrame({
    'year': [2024, 2024, 2024],
    'month': [1, 2, 4],
    'day': [1, 15, 14],
    'store': [1, 2, 1],
    'item': ['A', 'B', 'A'],
    'id': [101, 102, 103],
```

```
    'm1': [10, 15, 20],
    'm2': [5, 8, 12]
})

def is_holiday(x):
    india_holidays = holidays.CountryHoliday('IN')
    return 1 if x in india_holidays else 0

def which_day(row):
    d = datetime(row['year'], row['month'], row['day'])
    return d.weekday()

df['date'] = pd.to_datetime(df[['year', 'month', 'day']])
df['weekday'] = df.apply(which_day, axis=1)
df['is_weekend'] = df['date'].dt.dayofweek // 5  # 1 for Saturday or Sunday, 0 for
weekdays
df['is_holiday'] = df['date'].apply(is_holiday)

print(df.head())
from datetime import datetime
import holidays
import pandas as pd

# Assuming df is your DataFrame with 'year', 'month', 'day', and other columns
df = pd.DataFrame({
    'year': [2024, 2024, 2024],
    'month': [1, 2, 4],
    'day': [1, 15, 14],
    'store': [1, 2, 1],
    'item': ['A', 'B', 'A'],
    'id': [101, 102, 103],
    'm1': [10, 15, 20],
    'm2': [5, 8, 12]
})

def is_holiday(x):
    india_holidays = holidays.CountryHoliday('IN')
    return 1 if x in india_holidays else 0

def which_day(row):
    d = datetime(row['year'], row['month'], row['day'])
    return d.weekday()
```

```
df['date'] = pd.to_datetime(df[['year', 'month', 'day']])
df['weekday'] = df.apply(which_day, axis=1)
df['is_weekend'] = df['date'].dt.dayofweek
df['is_holiday'] = df['date'].apply(is_holiday)

print(df.head())
from datetime import datetime
import holidays
import pandas as pd
df = pd.DataFrame({
    'year': [2024, 2024, 2024],
    'month': [1, 2, 4],
    'day': [1, 15, 14],
    'store': [1, 2, 1],
    'item': ['A', 'B', 'A'],
    'id': [101, 102, 103],
    'm1': [10, 15, 20],
    'm2': [5, 8, 12]
})

def is_holiday(x):
    india_holidays = holidays.CountryHoliday('IN')
    return 1 if x in india_holidays else 0

def which_day(row):
    d = datetime(row['year'], row['month'], row['day'])
    return d.weekday()

df['date'] = pd.to_datetime(df[['year', 'month', 'day']])
df['weekday'] = df.apply(which_day, axis=1)
df['is_weekend'] = df['date'].dt.dayofweek
df['is_holiday'] = df['date'].apply(is_holiday)

print(df.head())
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df is your DataFrame with necessary columns including 'sales'
df = pd.DataFrame({
    'store': [1, 2, 1, 2, 1, 2],
    'year': [2024, 2024, 2024, 2024, 2024, 2024],
    'month': [1, 2, 3, 4, 5, 6],
    'weekday': [0, 1, 2, 3, 4, 5],  # Assuming 0=Monday, 1=Tuesday, ..., 5=Saturday
```

```
    'weekend': [0, 0, 0, 0, 0, 1],  # Assuming 0=Weekday, 1=Weekend
    'holidays': [0, 0, 1, 0, 0, 0],
    'sales': [100, 150, 120, 200, 180, 250]
})

features = ['store', 'year', 'month', 'weekday', 'weekend', 'holidays']

plt.subplots(figsize=(20, 10))
for i, col in enumerate(features):
    plt.subplot(2, 3, i + 1)
    df.groupby(col).mean()['sales'].plot.bar()
    plt.title(f'Mean Sales by {col}')

plt.tight_layout()
plt.show()
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df is your DataFrame with necessary columns including 'sales'
df = pd.DataFrame({
    'date': pd.date_range(start='2024-01-01', periods=30),
    'sales': [100, 150, 120, 200, 180, 250, 130, 170, 190, 220, 110, 130, 180, 210,
240,
              150, 160, 130, 170, 200, 120, 140, 180, 210, 230, 120, 140, 160, 180,
200],
    'weekday': pd.date_range(start='2024-01-01', periods=30).day_name()
})

plt.figure(figsize=(10, 5))
df.groupby('weekday').mean()['sales'].plot(marker='o', linestyle='-')
plt.title('Mean Sales by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Mean Sales')
plt.grid(True)
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Assuming df is your DataFrame with necessary columns including 'sales' and
'year'
df = pd.DataFrame({
    'year': [2024, 2024, 2024, 2024, 2024, 2024],
```

```
    'month': [1, 2, 3, 4, 5, 6],
    'weekday': [0, 1, 2, 3, 4, 5],
    'weekend': [0, 0, 0, 0, 0, 1],
    'holidays': [0, 0, 1, 0, 0, 0],
    'sales': [100, 150, 120, 200, 180, 250]
})

# Creating features DataFrame by dropping 'sales' and 'year' columns
features = df.drop(['sales', 'year'], axis=1)

# Creating target array using the 'sales' column
target = df['sales'].values

# Splitting the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(features, target, test_size=0.2,
random_state=42)

# Normalizing the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

print("Scaled Features (Training):")
print(X_train_scaled)
print("\nScaled Features (Validation):")
print(X_val_scaled)
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor, AdaBoostRegressor, BaggingRegressor
from sklearn.metrics import mean_absolute_error

# Assuming df is your DataFrame with necessary columns including 'sales' and
'year'
df = pd.DataFrame({
    'year': [2024, 2024, 2024, 2024, 2024, 2024],
    'month': [1, 2, 3, 4, 5, 6],
    'weekday': [0, 1, 2, 3, 4, 5],
    'weekend': [0, 0, 0, 0, 0, 1],
    'holidays': [0, 0, 1, 0, 0, 0],
    'sales': [100, 150, 120, 200, 180, 250]
```

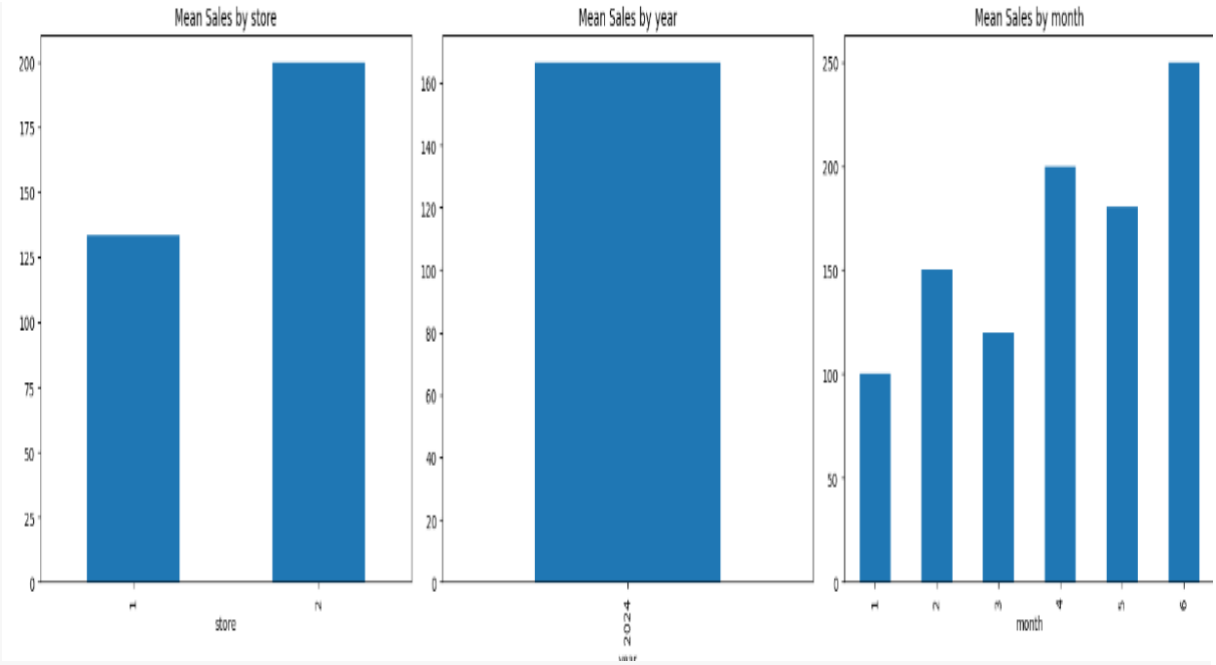
```
})

# Creating features DataFrame by dropping 'sales' and 'year' columns
features = df.drop(['sales', 'year'], axis=1)

# Creating target array using the 'sales' column
target = df['sales'].values

# Splitting the data into training and validation sets
X_train, X_val, Y_train, Y_val = train_test_split(features, target, test_size=0.2,
random_state=42)

# Normalizing the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)

# Defining models
models = [
    LinearRegression(),
    Ridge(),
    RandomForestRegressor(),
    GradientBoostingRegressor(),
    AdaBoostRegressor(),
    BaggingRegressor()
]

for model in models:
    model.fit(X_train, Y_train)

    print(f'Model: {model}')

    train_preds = model.predict(X_train)
    print('Training Error: ', mean_absolute_error(Y_train, train_preds))

    val_preds = model.predict(X_val)
    print('Validation Error: ', mean_absolute_error(Y_val, val_preds))

    print()
```

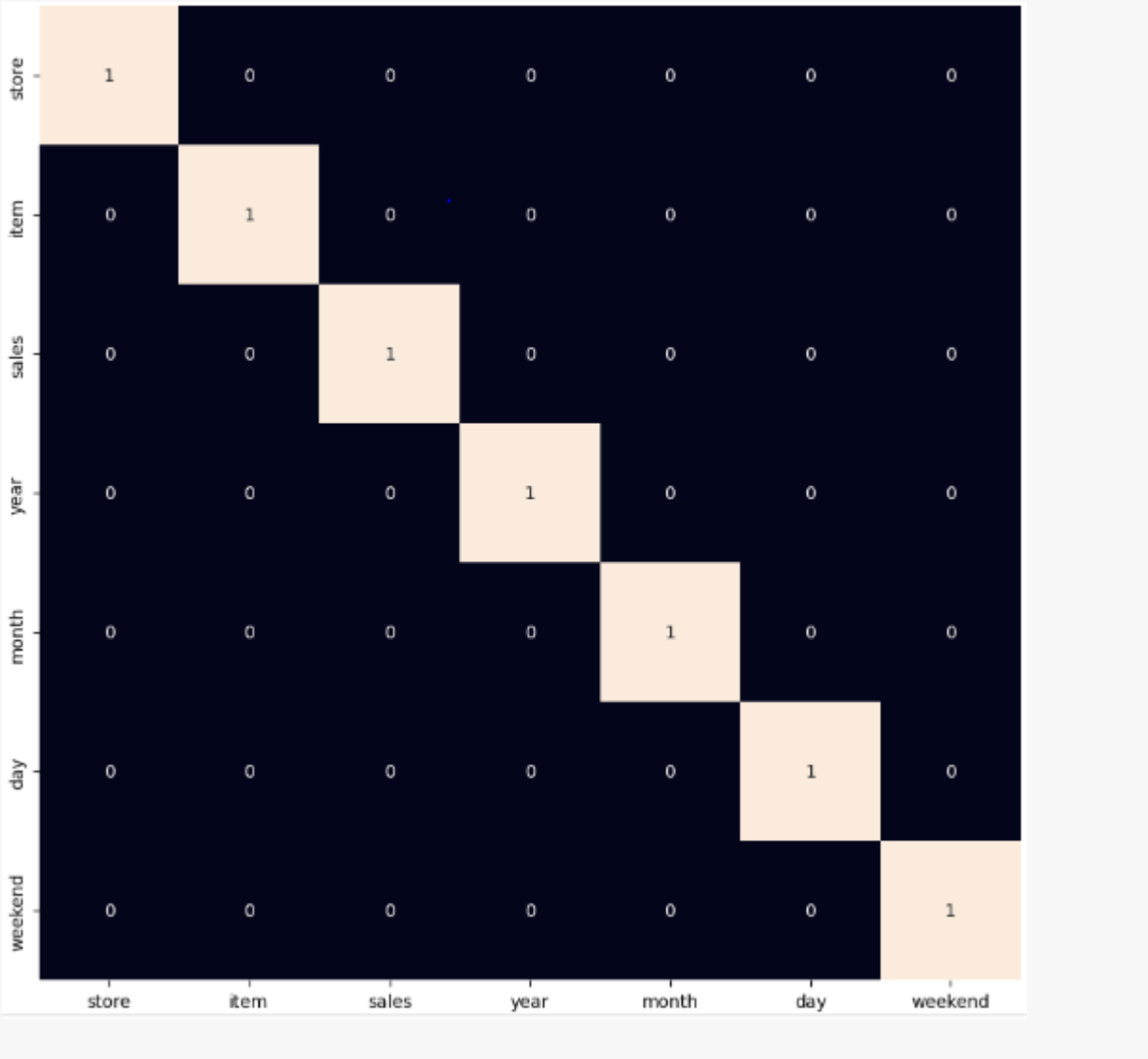
	date	store	item	sales
0	2013-01-01	1	1	13
1	2013-01-02	1	1	11
2	2013-01-03	1	1	14
3	2013-01-04	1	1	13
4	2013-01-05	1	1	10

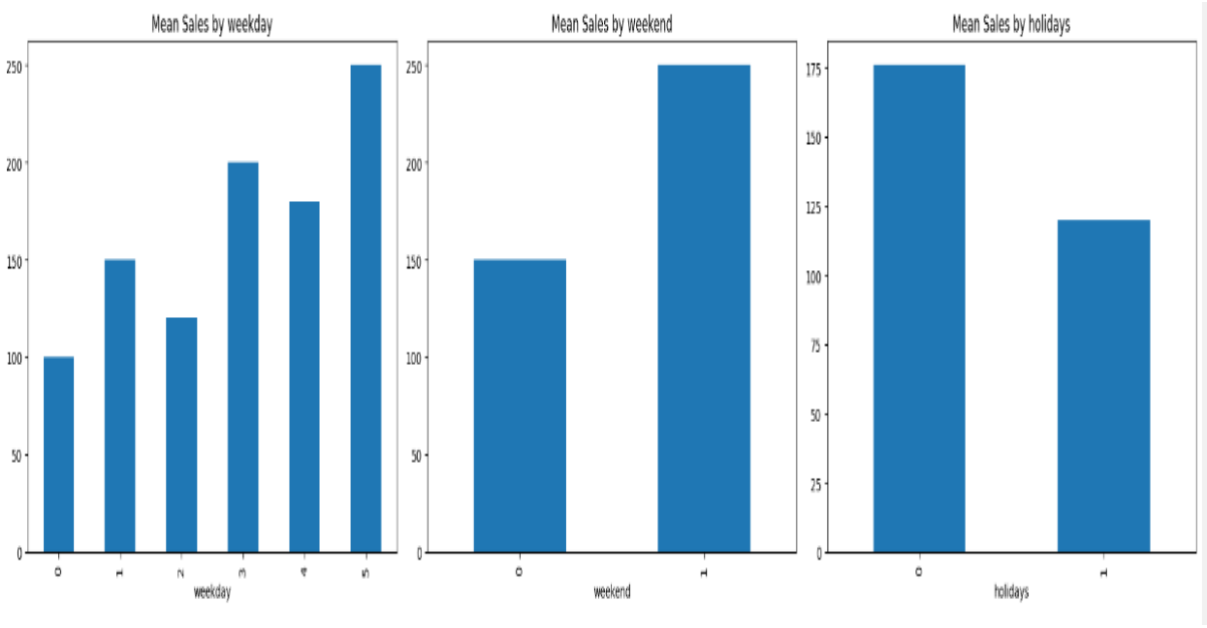
	date	store	item	sales
912995	2017-12-27	10	50	63
912996	2017-12-28	10	50	59
912997	2017-12-29	10	50	74
912998	2017-12-30	10	50	62
912999	2017-12-31	10	50	82

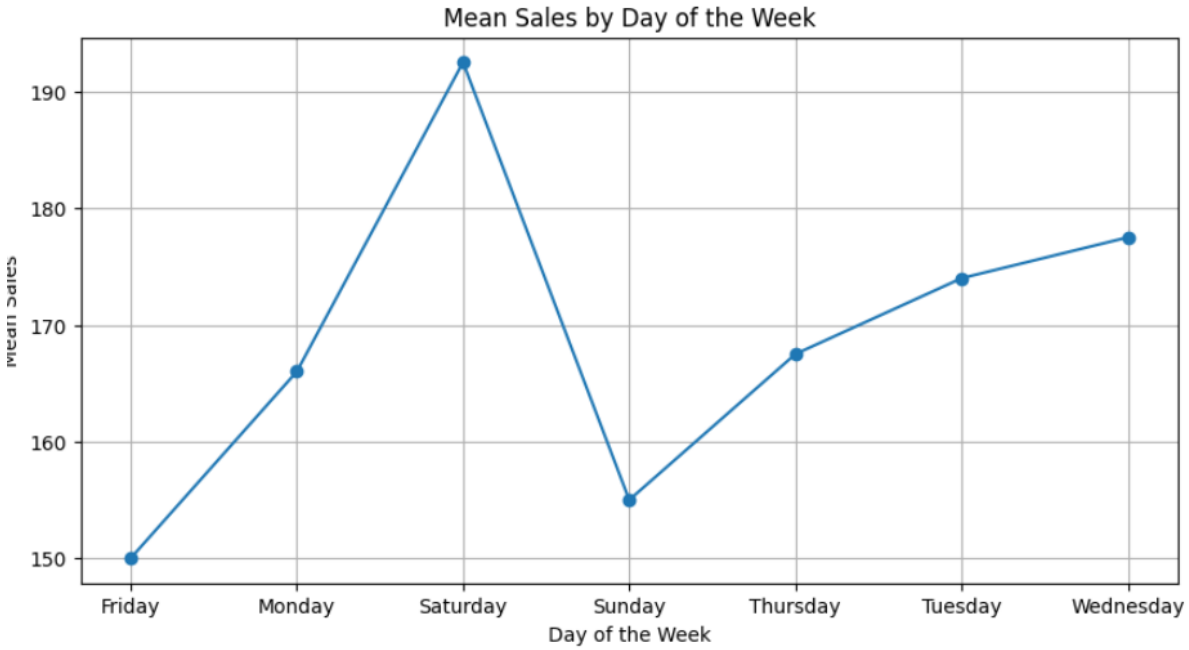
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 913000 entries, 0 to 912999
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    913000 non-null  object
 1   store   913000 non-null  int64
 2   item    913000 non-null  int64
 3   sales   913000 non-null  int64
dtypes: int64(3), object(1)
memory usage: 27.9+ MB
```

	store	item	sales
count	913000.000000	913000.000000	913000.000000
mean	5.500000	25.500000	52.250287
std	2.872283	14.430878	28.801144
min	1.000000	1.000000	0.000000
25%	3.000000	13.000000	30.000000
50%	5.500000	25.500000	47.000000
75%	8.000000	38.000000	70.000000


Mean Sales by store
store
Mean Sales by year
2024
Mean Sales by month
month


store
item
sales
year
month
day
weekend






```
Scaled Features (Training):
[[ 1.34164079  1.34164079  1.73205081 -0.57735027]
 [-1.34164079 -1.34164079 -0.57735027  1.73205081]
 [ 0.4472136   0.4472136  -0.57735027 -0.57735027]
 [-0.4472136  -0.4472136  -0.57735027 -0.57735027]]

Scaled Features (Validation):
[[-3.13049517 -3.13049517 -0.57735027 -0.57735027]
 [-2.23606798 -2.23606798 -0.57735027 -0.57735027]]
```

Model: LinearRegression()
Training Error: 1.4210854715202004e-14
Validation Error: 125.0000000000002

Model: Ridge()
Training Error: 11.621863799283148
Validation Error: 30.33968719452585

Model: RandomForestRegressor()
Training Error: 15.699999999999996
Validation Error: 34.30000000000001

Model: GradientBoostingRegressor()
Training Error: 0.000996052458294372
Validation Error: 25.0

Model: AdaBoostRegressor()
Training Error: 0.0
Validation Error: 55.0

Model: BaggingRegressor()
Training Error: 14.25
Validation Error: 39.0