```
import pandas as pd
import numpy as np

# Sample DataFrame
data = {
    'date': pd.date_range(start='1/1/2020', periods=100, freq='D'),
    'value1': np.random.rand(100),
    'value2': np.random.rand(100)
}
df = pd.DataFrame(data)
# Extracting date features
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
df['day_of_week'] = df['date'].dt.dayofweek

# Creating interaction features
df['value1_squared'] = df['value1'] ** 2
df['value1_value2_interaction'] = df['value1'] * df['value2']

# Aggregation example
df['value1_mean'] = df['value1'].rolling(window=7).mean()
df['value2_sum'] = df['value2'].rolling(window=7).sum()
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Dropping the date column and any non-numeric columns
X = df.drop(columns=['date'])

# Filling NaN values
X = X.fillna(0)

# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Applying PCA
pca = PCA(n_components=5)
X_pca = pca.fit_transform(X_scaled)

# Checking explained variance
explained_variance = pca.explained_variance_ratio_
print(explained_variance)
from sklearn.ensemble import RandomForestRegressor

# Assuming a target variable y
y = np.random.rand(100)

# Training the model
model = RandomForestRegressor()
model.fit(X, y)

# Getting feature importances
importances = model.feature_importances_
feature_importances = pd.Series(importances, index=X.columns).sort_values(ascending=False)
print(feature_importances)
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

# Using RFE with Linear Regression
model = LinearRegression()
rfe = RFE(model, n_features_to_select=5)
rfe = rfe.fit(X, y)

# Selected features
selected_features = X.columns[rfe.support_]
print(selected_features)
# Using the selected features from RFE
X_selected = X[selected_features]

# Train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)

# Training the model with selected features
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluating the model
from sklearn.metrics import mean_squared_error
y_pred = model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:",mse)
```

```
[0.28966432 0.20353184 0.16419955 0.13113033 0.10945282]
value2                        0.196385
value2_sum                    0.188045
day                           0.130407
value1_value2_interaction     0.125055
value1_mean                   0.113143
day_of_week                   0.081737
value1_squared                0.068531
value1                        0.066968
month                         0.029729
year                          0.000000
dtype: float64
Index(['value1', 'value1_squared', 'value1_value2_interaction', 'value1_mean',
       'value2_sum'],
      dtype='object')
Mean Squared Error: 0.07993403212780484
```