Name: Chinmay Mhatre Roll No 22102B2001 BE CMPN B Github link: https://github.com/chinmay0910/ML-Lab---VIT

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns
```

```
!pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.7-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2.1.4)
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2024.8.30)
Requirement already satisfied: numpy<2,>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucir
Downloading ucimlrepo-0.0.7-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.7
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
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+compres
                            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
```

```
data = pd.concat([X, y], axis=1)
```

```
data.head()
data.isnull().sum()
```

```
data.shape
```

```
(1030, 9)
```

```
plt.figure(figsize=(8, 5))
sns.histplot(data['Concrete compressive strength'], bins=30, kde=True)
plt.title('Distribution of Concrete Compressive Strength')
plt.show()
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Concrete Compressive Strength Features')
plt.show()
plt.figure(figsize=(8, 5))
sns.scatterplot(x=data['Cement'], y=data['Concrete compressive strength'])
plt.title('Cement Content vs Compressive Strength')
plt.show()
```

```
X.head()
```

```
y
```

```
#Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#Standardising the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
svm_regressor = SVR()
```

```
svm_regressor.fit(X_train, y_train)
```

```
y_pred = svm_regressor.predict(X_test)
```

```
y_pred
```

```
array([42.02837189, 53.07178175, 55.4244467 , 52.25058179, 27.15432674,
       41.22365799, 26.95118384, 47.40291511, 31.32277358, 43.28318885,
       34.6409369 , 20.64832546, 46.85470404, 44.67936286, 26.7697804 ,
       34.53273197, 31.99436603, 25.82984409, 33.88688392, 27.83015034,
       32.0766513 , 37.40398295, 41.39953655, 26.31465017, 31.75662963,
       35.14831839, 16.7140779 , 45.70059883, 44.53785265, 22.78996674,
       37.58345848, 33.15714416, 39.86020844, 46.99425817, 25.63722485,
       35.14087252, 30.44223862, 41.21976612, 21.01493474, 41.87018131,
       19.44796208, 19.65825568, 33.41468603, 47.23060742, 19.46405782,
       61.1972441 , 42.85999254, 39.01843491, 22.32984782, 21.63059085,
       42.85887622, 40.46806868, 30.82688732, 28.15760928, 42.06606084,
       51.34657107, 28.70469115, 17.85449175, 34.87772226, 21.08940793,
       40.98866094, 26.54925385, 40.0213611 , 55.33368625, 23.88083109,
       26.88969139, 34.53063954, 22.45847868, 28.91575364, 24.3361942 ,
       20.9310734 , 20.10855879, 15.28413169, 34.59239126, 28.7671655 ,
       18.49734528, 42.45549697, 48.17558871, 42.4804513 , 19.89836997,
       42.30548047, 48.05048053, 37.13820692, 32.84148272, 41.43166993,
       52.25058179, 32.83942845, 33.66063211, 25.3157254 , 21.52790772,
       28.19085024, 62.82200304, 24.00379327, 44.59481214, 36.58371711,
       42.68775946, 27.52852521, 27.60116142, 30.11546533, 36.08876735,
       30.35783567, 39.91181395, 41.91213076, 28.87319528, 48.72032177,
       17.37285367, 44.583502  , 34.0371043 , 31.93585182, 48.53510181,
       39.34507836, 42.45748984, 33.38086995, 41.39399152, 39.03262379,
       55.17943086, 23.90129334, 34.99041144, 41.80943612, 36.65042707,
       27.99859604, 29.89665989, 37.56883452, 32.91376618, 27.86436518,
       35.15561274, 44.01525996, 52.90827295, 40.62731203, 34.34042727,
       17.72013924, 33.75971214, 19.78318523, 58.07656691, 21.98925925,
       42.66326857, 23.75930396, 41.91729814, 30.02926941, 36.58780588,
       29.20732677, 35.55921867, 24.82304432, 32.44637406, 43.44676723,
       36.87214701, 31.02373423, 24.40378311, 20.54619995, 20.09422917,
       36.6468206 , 28.56065734, 42.59756949, 26.92409654, 44.81079565,
       39.92902934, 31.49325547, 48.94610101, 53.07178175, 45.78797978,
       16.4144045 , 46.74858406, 31.94937721, 43.8093915 , 52.35562678,
       48.02848332, 41.93868519, 24.04958892, 46.99949428, 32.23050546,
       35.60120172, 24.83537841, 24.13891503, 35.29104136, 35.72563098,
       17.7927775 , 21.58176276, 42.14097109, 46.34091142, 45.90716998,
       32.41473521, 25.70496622, 48.54746054, 45.6671333 , 35.85312305,
       44.30182973, 22.9703455 , 32.32878459, 46.7212128 , 18.44998887,
       41.82270862, 41.18892337, 43.68048851, 41.8764995 , 37.72531942,
       27.96939711, 40.64346143, 28.6264187 , 26.06232217, 24.44846415,
       39.94780809, 55.17943086, 24.14668569, 46.94010243, 47.44810071,
       39.04886052])
```

```
mse = mean_squared_error(y_test, y_pred)


r2 = r2_score(y_test, y_pred)


print(f"Mean Squared Error: {mse:.4f}")
print(f"R-squared: {r2:.4f}")
```

```
Mean Squared Error: 88.9594
R-squared: 0.6548
```

```
plt.scatter(y_test, y_pred, color='blue', edgecolor='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()
```

Start coding or generate with AI.