```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline


# Load the dataset
df = pd.read_csv('winequality-red.csv', sep=',', encoding='cp1252')


df.info()
df.head()


# Remove any missing values
df = df.dropna()

# Separate the features and the target variable
X = df.drop(['quality'], axis=1)
Y = df['quality']

# checking the distribution of features
df.describe()


plt.figure(figsize=(16,10))
sns.heatmap(df.corr(),annot=True, annot_kws={"size": 14})
sns.set_style('white')
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
# Split the data into training and test samples
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
print(f'Sample size for training: {X_train.shape}, {Y_train.shape}')
print(f'Sample size for testing: {X_test.shape}, {Y_test.shape}')
```

```
# Linear univariate regression
model = LinearRegression()
# Навчання моделі
model.fit(X_train.iloc[:, -1:], Y_train)
# Прогнозування для тестової вибірки
y_pred = model.predict(X_test.iloc[:, -1:])


mse = mean_squared_error(Y_test, y_pred)
accuracy = model.score(X_test.iloc[:, -1:], Y_test)
print("Linear univariate regression:")
print("MSE = {:.4f}".format(mse), "\nAccuracy = {:.4f}".format(accuracy))

plt.xlabel('Alcohol')
plt.ylabel('Quality')
plt.title('Linear univariate regression')

plt.scatter(X_test.iloc[:, -1:], Y_test, alpha=0.7, color='blue')
plt.show()
# Linear multivariate regression
model = LinearRegression()
# Навчання моделі
model.fit(X_train, Y_train)
# Прогнозування для тестової вибірки
y_pred = model.predict(X_test)

mse = mean_squared_error(Y_test, y_pred)
accuracy = model.score(X_test, Y_test)
print("Linear multivariate regression:")
print("MSE = {:.4f}".format(mse), "\nAccuracy = {:.4f}".format(accuracy))

plt.scatter(Y_test, y_pred, color='blue')
plt.xlabel('Valid values')
plt.ylabel('Predicted values')
plt.title('Linear multivariate regression')
plt.show()


# Polynomial regression with degree 2
poly = PolynomialFeatures(degree=2)
```

```
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model_poly = LinearRegression()
model_poly.fit(X_train_poly, Y_train)

y_pred = model_poly.predict(X_test_poly)

mse = mean_squared_error(Y_test, y_pred)
accuracy = model_poly.score(X_test_poly, Y_test)
print("Polynomial regression with degree 2:")
print("MSE = {:.4f}".format(mse), "\nAccuracy = {:.4f}".format(accuracy))

plt.scatter(Y_test, y_pred, color='blue')
plt.xlabel('Valid values')
plt.ylabel('Predicted values')
plt.title('Polynomial regression with degree 2')
plt.show()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

df = pd.read_csv('Data4.csv', sep=';', decimal=',', encoding='windows-1251').rename(columns={'Unnamed: 0':
'Country'})
df.info()
print(df.head())
df.corr()


pd.plotting.scatter_matrix(df, figsize=(10, 10))
plt.show()


Y = df['Cql']

l_model1 = LinearRegression().fit(df[['le', 'lec', 'ls']], Y)
l_model2 = LinearRegression().fit(df[['lec', 'ls']], Y)
l_model3 = LinearRegression().fit(df[['le', 'ls']], Y)
l_model4 = LinearRegression().fit(df[['le', 'lec']], Y)
l_model5 = LinearRegression().fit(df['le'].to_numpy().reshape(-1, 1), Y)
l_model6 = LinearRegression().fit(df['lec'].to_numpy().reshape(-1, 1), Y)
l_model7 = LinearRegression().fit(df['ls'].to_numpy().reshape(-1, 1), Y)


p_model1 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model1.fit(df[['le', 'lec', 'ls']], Y)

p_model2 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model2.fit(df[['lec', 'ls']], Y)

p_model3 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model3.fit(df[['le', 'lec']], Y)

p_model4 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model4.fit(df['lec'].to_numpy().reshape(-1, 1), Y)
```

```
linear_models = [l_model5, l_model6, l_model7]

params = ['le', 'lec', 'ls']
params_values = []
y_pred = []

for i in range(len(params)):
    values = np.linspace(df[params[i]].min(), df[params[i]].max()).reshape(-1, 1)
    params_values.append(values)
    y_pred.append(linear_models[i].predict(values))

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

for i in range(len(axes)):
    axes[i].set_title(f'Лінійна регресія Cql ~ {params[i]}')
    axes[i].set_xlabel(params[i])
    axes[i].set_ylabel('Cql')
    axes[i].grid(linestyle='--')
    axes[i].scatter(df[params[i]], Y)
    axes[i].plot(params_values[i], y_pred[i], color='red')


X_pol_test = np.linspace(df['lec'].min(), df['lec'].max()).reshape(-1, 1)
Y_pol_pred = p_model4.predict(X_pol_test)

plt.figure(figsize=(5, 5))

plt.title('Поліноміальна регресія Cql ~ lec')
plt.xlabel('Cql')
plt.ylabel('lec')
plt.grid(linestyle='--')

plt.scatter(df['lec'], Y)
plt.plot(X_pol_test, Y_pol_pred, color='red')

plt.show()
%matplotlib inline
```

```
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

X_3d = params_values[0]
Y_3d = params_values[1]

XX, YY = np.meshgrid(X_3d, Y_3d)

Z = []
for i in range(len(Y_3d)):
    temp = []
    for j in range(len(X_3d)):
        temp.append(p_model3.predict(np.array([X_3d[j], Y_3d[i]]).T)[0])
    Z.append(temp)

Z = np.array(Z)

ax.set_title('Поліноміальна регресія Cql ~ Ie, Iec')
ax.set_xlabel('Ie')
ax.set_ylabel('Iec')
ax.set_zlabel('Cql')
ax.plot_surface(
    XX, YY,
    np.array(Z),
    color='green',
    alpha=0.5
)
ax.scatter(df['Ie'], df['Iec'], Y)

plt.show()

df_test = pd.read_csv('Data4t.csv', encoding='windows-1251', sep=';', decimal=',').rename(columns={'Unnamed:
0': 'Country'})

df_test
test_predictions = []
names = ['l_model1' ,'l_model2' ,'l_model3' ,'l_model4' ,'l_model5' ,'l_model6' ,'l_model7' , 'p_model1', 'p_mode2',
'p_model3', 'p_model4']

test_predictions.append(l_model1.predict(df_test[['Ie', 'Iec', 'Is']]))
```

```
test_predictions.append(l_model2.predict(df_test[['lec', 'ls']]))
test_predictions.append(l_model3.predict(df_test[['le', 'ls']]))
test_predictions.append(l_model4.predict(df_test[['le', 'lec']]))
test_predictions.append(l_model5.predict(df_test['le'].to_numpy().reshape(-1, 1)))
test_predictions.append(l_model6.predict(df_test['lec'].to_numpy().reshape(-1, 1)))
test_predictions.append(l_model7.predict(df_test['ls'].to_numpy().reshape(-1, 1)))
test_predictions.append(p_model1.predict(df_test[['le', 'lec', 'ls']]))
test_predictions.append(p_model2.predict(df_test[['lec', 'ls']]))
test_predictions.append(p_model3.predict(df_test[['le', 'lec']]))
test_predictions.append(p_model4.predict(df_test['lec'].to_numpy().reshape(-1, 1)))

test_predictions = np.array(test_predictions)

best = np.sum((test_predictions - df_test['Cql'].to_numpy()) ** 2, axis=1).argmin()

print(f'The best solusion is {names[best]}')
```