Assignment 4 problem 1

```
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import root_mean_squared_error, r2_score

# Load the diabetes dataset
diabetes = load_diabetes()
data = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
target = diabetes.target

# Base variables: bmi and s5
base_features = ['bmi', 's5', 'bp']
rmse_results = []

# Iterate through other features
for feature in ['s3', 's6', 's1', 's2', 's4']:
  # Combine base features with the current feature
  selected_features = base_features + [feature]

  # Prepare data
  X = data[selected_features]
  X_train, X_test, y_train, y_test = train_test_split(X, target, test_size=0.2,
random_state=42)

  # Train model
  model = LinearRegression()
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)

  # Compute RMSE
  rmse = root_mean_squared_error(y_test, y_pred)
  r2 = r2_score(y_test, y_pred)
  rmse_results.append({
    'Feature Added': feature,
    'RMSE': rmse,
    'R² Score': r2
  })
```

```
# Create a DataFrame to compare results
rmse_df = pd.DataFrame(rmse_results)

print(rmse_df)

"""
```

a) Which variable would you add next? Why?
After BMI and S5, we added all other attributes of the dataset namely Bp, S1, S2, S3, S4, and S6. We focused the analysis on Root Mean Squarred Error (RMSE),
and R^2 Score. We calculated the values and found that addition of BP resulted in lowering of RMSE. Thus, we decided to select BP as the next attribute to be added.

Feature Added	RMSE	R^2 Score
bp	53.768366	0.454331
s3	53.705538	0.455605
s6	53.810247	0.453481
s1	54.221504	0.445095
s2	54.090240	0.447778
s4	53.801874	0.453651

b) How does adding it affect the model's performance? Compute metrics and compare to having just bmi and s5.
We observed that addition of BP reduced the Root Mean Squarred Error (RMSE). With less error, the prediction power of the model will improve.

Model	RMSE	R^2 Score
Base (bmi, s5)	53.868701	0.452293
Base + bp	53.768366	0.454331

c) Does it help if you add even more variables?
To test this, we considered bmi + s5 + bp as the new base and then further added other features one by one.

Model	RMSE	R^2 Score
s3	54.127467	0.447018
s6	53.779654	0.454102
s1	54.180881	0.445926
s2	53.886656	0.451927
s4	53.854365	0.452584

What we found that the next attribute that can be added is S6. However, S6 causes the RMSE to increase. Thus, we will not add any more attributes after bp.

Model	RMSE	R^2 Score
bmi, s5, bp	53.768366	0.454331
bmi, s5, bp, s6	53.779654	0.454102

"""

Assignment 4 problem 2

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import root_mean_squared_error, r2_score

# Step 0: Load dataset
file_path = "C:\\Users\\ prashanth\\Downloads\\50_Startups.csv"  # Replace with
correct file path
data = pd.read_csv(file_path)

# Encode the State column using One-Hot Encoding
data_encoded = pd.get_dummies(data, columns=['State'], drop_first=True)

# Step 1: Identify variables
print(data_encoded.info())

# Step 2: Correlation Analysis
correlation_matrix = data_encoded.corr()
print("Correlation Matrix:")
print(correlation_matrix)

# Step 3: Variable Selection
selected_features = ['R&D Spend', 'Marketing Spend']  # High correlation with Profit
print("Selected Predictors:", selected_features)

# Step 4: Plot explanatory variables against profit
for feature in selected_features:
   plt.scatter(data_encoded[feature], data_encoded['Profit'], alpha=0.6)
   plt.title(f"{feature} vs Profit")
   plt.xlabel(feature)
   plt.ylabel("Profit")
```

```
    plt.grid(True)
    plt.show()

# Step 5: Data Splitting
X = data_encoded[selected_features]
y = data_encoded['Profit']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Model Training
model = LinearRegression()
model.fit(X_train, y_train)

# Step 7: Performance Evaluation
# Training Data
y_train_pred = model.predict(X_train)
rmse_train = root_mean_squared_error(y_train, y_train_pred)
r2_train = r2_score(y_train, y_train_pred)

# Testing Data
y_test_pred = model.predict(X_test)
rmse_test = root_mean_squared_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)

# Display Results
print("Training Data Metrics:")
print(f"RMSE: {rmse_train}, R²: {r2_train}")
print("Testing Data Metrics:")
print(f"RMSE: {rmse_test}, R²: {r2_test}")
```

Assignment 4 problem 3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import r2_score

# Step 1: Read the dataset
file_path = "C:\\Users\\prashanth\\Downloads\\Auto.csv"
```

```
data = pd.read_csv(file_path)

# Inspect the dataset structure
data.info()

"""
Step 2: Setup Regression Variables
- Target variable: 'mpg' (miles per gallon)
- Predictor variables: Exclude 'mpg', 'name', and 'origin'
"""
X = data.drop(columns=['mpg', 'name', 'origin'])
y = data['mpg']

# Remove missing values if present
X = X.dropna()
y = y[X.index]

# Step 3: Split data into training and testing sets (80/20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

"""
Step 4-5: Implement Ridge and LASSO regression for various alpha values and find
optimal alpha
"""
alphas = np.logspace(-3, 3, 100)  # Generate alpha values from 0.001 to 1000
ridge_scores = []
lasso_scores = []

for alpha in alphas:
    # Ridge regression
    ridge_model = Ridge(alpha=alpha)
    ridge_model.fit(X_train, y_train)
    ridge_scores.append(r2_score(y_test, ridge_model.predict(X_test)))

    # Lasso regression
    lasso_model = Lasso(alpha=alpha, max_iter=10000)
    lasso_model.fit(X_train, y_train)
    lasso_scores.append(r2_score(y_test, lasso_model.predict(X_test)))

# Step 6: Plot R2 scores as functions of alpha
plt.figure(figsize=(10, 6))
```

```
plt.plot(alphas, ridge_scores, label='Ridge R²', color='blue', marker='o', markersize=4)
plt.plot(alphas, lasso_scores, label='Lasso R²', color='red', marker='o', markersize=4)
plt.xscale('log')  # Log scale for alpha
plt.xlabel('Alpha')
plt.ylabel('R² Score')
plt.title('R² Scores for Ridge and Lasso Regression')
plt.legend()
plt.grid(True)
plt.show()

"""
Step 7: Identify optimal alpha
Find the alpha value that gives the maximum R² score for each regressor.
"""
optimal_ridge_alpha = alphas[np.argmax(ridge_scores)]
optimal_lasso_alpha = alphas[np.argmax(lasso_scores)]
optimal_ridge_score = max(ridge_scores)
optimal_lasso_score = max(lasso_scores)

# Results
results = pd.DataFrame({
    'Regressor': ['Ridge', 'Lasso'],
    'Optimal Alpha': [optimal_ridge_alpha, optimal_lasso_alpha],
    'Best R² Score': [optimal_ridge_score, optimal_lasso_score]
})

# Print optimal alpha values and corresponding R² scores
print("Optimal Ridge Regression Alpha and R²:")
print(f"Alpha: {optimal_ridge_alpha}, Best R² Score: {optimal_ridge_score}")

print("\nOptimal Lasso Regression Alpha and R²:")
print(f"Alpha: {optimal_lasso_alpha}, Best R² Score: {optimal_lasso_score}")

# Display results in tabular format
print("\nComparison Table:")
print(results)

# If running locally, the results can also be written to a file
results.to_csv("optimal_alpha_results.csv", index=False)

plt.figure(figsize=(10, 6))
```

```
plt.plot(alphas, ridge_scores, label='Ridge R²', color='blue', marker='o', markersize=4)
plt.plot(alphas, lasso_scores, label='Lasso R²', color='red', marker='o', markersize=4)
plt.xscale('log')  # Log scale for alpha
plt.xlabel('Alpha')
plt.ylabel('R² Score')
plt.title('R² Scores for Ridge and Lasso Regression')
plt.legend()
plt.grid(True)
plt.show()
```