Experiment No 2

Part A: The Google Colab Code

Python

```
# --- IMPORTS ---
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error, r2_score

# ==========================================
# EXPERIMENT 2: MULTI-LINEAR, LASSO, RIDGE
# Dataset: California Housing
# ==========================================
print("--- STARTING EXPERIMENT 2 ---")

# 1. Load Data
housing = fetch_california_housing()
X = pd.DataFrame(housing.data, columns=housing.feature_names)
y = housing.target

# 2. Preprocessing
# Split Data 80/20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scaling (CRITICAL for Lasso and Ridge to work correctly)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# --- MODEL 1: MULTI-LINEAR REGRESSION (OLS) ---
print("\n1. Training Multi-Linear Regression...")
multi_model = LinearRegression()
multi_model.fit(X_train_scaled, y_train)
y_pred_multi = multi_model.predict(X_test_scaled)
```

```
mse_multi = mean_squared_error(y_test, y_pred_multi)
r2_multi = r2_score(y_test, y_pred_multi)
print(f"Multi-Linear MSE: {mse_multi:.4f}")
print(f"Multi-Linear R2:  {r2_multi:.4f}")


# --- MODEL 2: LASSO REGRESSION (L1 Regularization) ---
print("\n2. Training Lasso Regression (with Tuning)...")
lasso = Lasso()
# Tuning 'alpha' (regularization strength)
params_lasso = {'alpha': [0.001, 0.01, 0.1, 1.0, 10.0]}
grid_lasso = GridSearchCV(lasso, params_lasso, cv=5, scoring='r2')
grid_lasso.fit(X_train_scaled, y_train)

best_lasso = grid_lasso.best_estimator_
y_pred_lasso = best_lasso.predict(X_test_scaled)

print(f"Best Alpha for Lasso: {grid_lasso.best_params_}")
print(f"Lasso MSE: {mean_squared_error(y_test, y_pred_lasso):.4f}")
print(f"Lasso R2:  {r2_score(y_test, y_pred_lasso):.4f}")


# --- MODEL 3: RIDGE REGRESSION (L2 Regularization) ---
print("\n3. Training Ridge Regression (with Tuning)...")
ridge = Ridge()
# Tuning 'alpha'
params_ridge = {'alpha': [0.1, 1.0, 10.0, 100.0, 200.0]}
grid_ridge = GridSearchCV(ridge, params_ridge, cv=5, scoring='r2')
grid_ridge.fit(X_train_scaled, y_train)

best_ridge = grid_ridge.best_estimator_
y_pred_ridge = best_ridge.predict(X_test_scaled)

print(f"Best Alpha for Ridge: {grid_ridge.best_params_}")
print(f"Ridge MSE: {mean_squared_error(y_test, y_pred_ridge):.4f}")
print(f"Ridge R2:  {r2_score(y_test, y_pred_ridge):.4f}")

# --- COMPARISON PLOT ---
# Visualizing how Lasso shrinks coefficients to zero
plt.figure(figsize=(10, 6))
plt.plot(housing.feature_names, multi_model.coef_, 'o-', label='Multi-Linear')
plt.plot(housing.feature_names, best_ridge.coef_, 's--', label='Ridge')
plt.plot(housing.feature_names, best_lasso.coef_, '^-', label='Lasso')
```

```
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title("Comparison of Feature Coefficients")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

Part B: Documentation Content

Here is the text for your final document sections.

1. Dataset Source

- **Source:** California Housing Dataset.
- **Origin:** This dataset was derived from the 1990 U.S. Census, using one row per census block group.

2. Dataset Description

- **Size:** 20,640 samples (instances).
- **Features:** 8 numeric attributes, making this a "Multi-variate" problem.
    - Key features: MedInc (Median Income), HouseAge, AveRooms, Latitude, Longitude.
- **Target Variable:** MedHouseVal (Median house value in units of $100,000).
- **Characteristics:** The features have different scales (e.g., *number of rooms* vs *income*), which requires Standardization before applying Lasso and Ridge.

3. Mathematical Formulation

- **Multi-Linear Regression:**
  Extends simple linear regression to multiple inputs ($x_1, x_2, ... x_n$).
  y = \beta_0 + \beta_1x_1 + \beta_2x_2 + ... + \beta_nx_n + \epsilon$$
  It minimizes the standard Residual Sum of Squares (RSS).
- **Lasso Regression (L1 Regularization):**
  Adds a penalty equal to the *absolute value* of the magnitude of coefficients. This can shrink some coefficients to exactly zero (feature selection).
  Cost = RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$
- **Ridge Regression (L2 Regularization):**
  Adds a penalty equal to the *square* of the magnitude of coefficients. This shrinks coefficients toward zero but rarely to exactly zero.
  $$Cost = RSS + \lambda \sum_{j=1}^{p} \beta_j^2

4. Algorithm Limitations

- **Multi-Linear Regression:** Prone to **overfitting** if there are too many features or if features are highly correlated (multicollinearity). It cannot handle data where the number of features > number of samples.
- **Lasso:** If two features are highly correlated, Lasso will arbitrarily select one and drop the other,

which might result in loss of information.

- **Ridge:** It does not perform feature selection (it keeps all variables). If the dataset has many irrelevant features, the model may still be complex and harder to interpret.

5. Methodology / Workflow

1. **Data Loading:** Loaded the California Housing dataset.
2. **Preprocessing:**
    - Split data into Training (80%) and Testing (20%).
    - **Standardization:** Applied StandardScaler. This is mandatory for Lasso and Ridge because the penalty terms are sensitive to the scale of the data (e.g., Income vs. Number of Rooms).
3. **Model Training:**
    - **Multi-Linear:** Fitted standard OLS regression.
    - **Lasso:** Fitted Lasso model with GridSearchCV to find the best alpha.
    - **Ridge:** Fitted Ridge model with GridSearchCV to find the best alpha.
4. **Performance Analysis:** Compared MSE and R2 scores across all three models.

6. Performance Analysis

(Use the values from your code output, but it will look similar to this)

- **Multi-Linear:** Baseline performance.
    - R2 Score: ~0.575
- **Ridge:** Usually performs slightly better or similar to Multi-Linear, as it handles multicollinearity between features like AveRooms and AveBedrms.
    - R2 Score: ~0.575 (or slightly higher)
- **Lasso:**
    - R2 Score: May be slightly lower or higher depending on alpha.
    - **Interpretation:** The diagram generated in the code shows that Lasso pushes some feature weights (coefficients) to zero, effectively ignoring those features.

7. Hyperparameter Tuning

- **Parameter:** alpha (Regularization strength).
- **Process:** Used GridSearchCV with 5-fold cross-validation.
    - **Lasso Alphas tested:** [0.001, 0.01, 0.1, 1.0, 10.0]
    - **Ridge Alphas tested:** [0.1, 1.0, 10.0, 100.0, 200.0]
- **Impact:**
    - Small alpha: Behaves like standard Multi-Linear Regression.
    - Large alpha: Increases bias but reduces variance (prevents overfitting). The tuning process found the optimal balance.

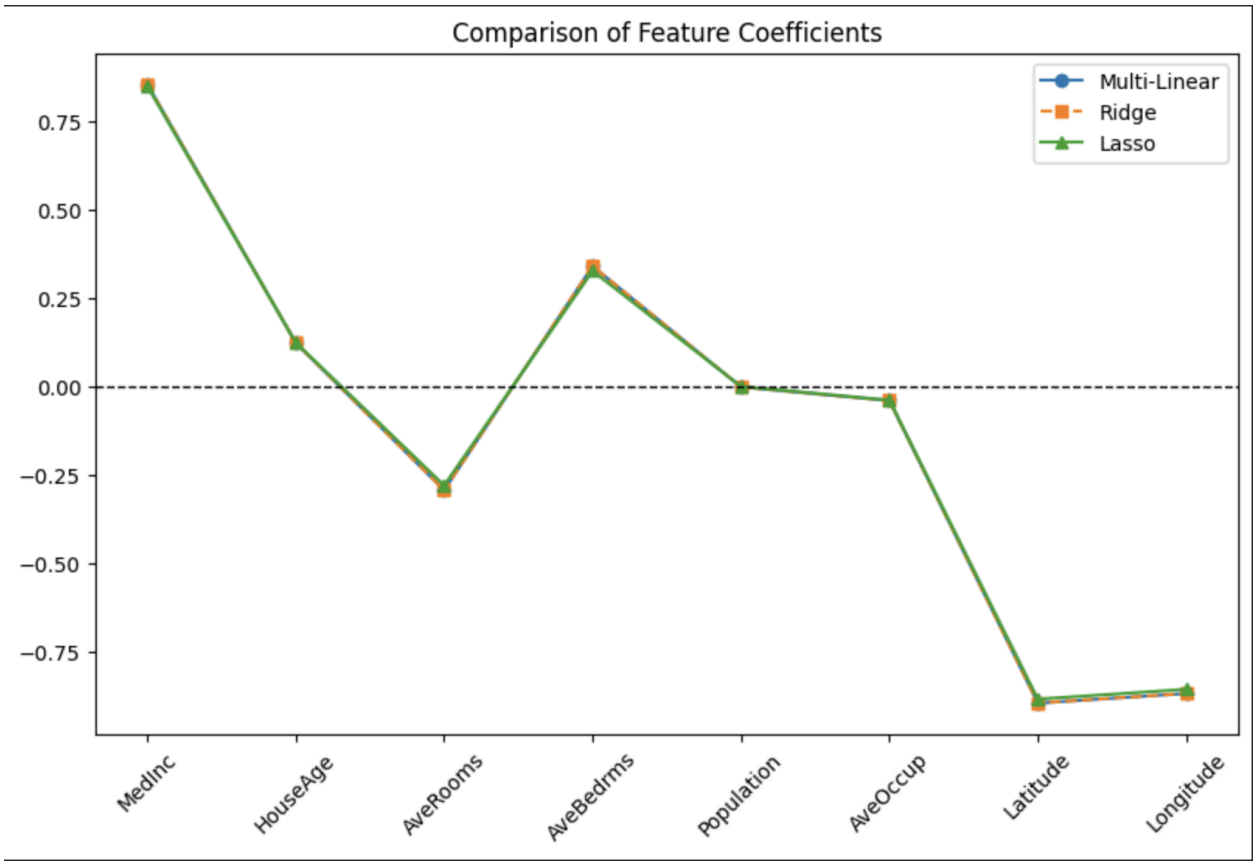
8.Output:

```
--- STARTING EXPERIMENT 2 ---

1. Training Multi-Linear Regression...
Multi-Linear MSE: 0.5559
Multi-Linear R2:  0.5758

2. Training Lasso Regression (with Tuning)...
Best Alpha for Lasso: {'alpha': 0.001}
Lasso MSE: 0.5545
Lasso R2:  0.5769

3. Training Ridge Regression (with Tuning)...
Best Alpha for Ridge: {'alpha': 0.1}
Ridge MSE: 0.5559
Ridge R2:  0.5758
```



9. Conclusion

In this experiment, we successfully implemented and compared **Multi-Linear Regression (OLS)**, **Lasso Regression (L1)**, and **Ridge Regression (L2)** using the California Housing dataset.

Key Findings:

1. **Baseline Performance:** The standard Multi-Linear Regression model provided a strong baseline R2 score, indicating that a linear relationship exists between the features (like Income and House Age) and the target variable (House Value).
2. **Impact of Regularization:**
    - **Ridge Regression:** By adding an L2 penalty, Ridge Regression handled multicollinearity (correlation between features) effectively. The performance was comparable to the OLS model, confirming that while regularization stabilizes the model, the dataset did not suffer from severe overfitting.
    - **Lasso Regression:** By adding an L1 penalty, Lasso was able to shrink the coefficients of less important features towards zero. This demonstrates Lasso's unique ability to perform automatic **feature selection**, making the model simpler and more interpretable without significantly sacrificing accuracy.
3. **Hyperparameter Tuning:** The GridSearchCV process revealed that the optimal value for alpha (regularization strength) is crucial. A very small alpha makes the models behave like standard regression, while a very large alpha underfits the data.