Lab 7A – L2 Regularization (Ridge Regression)

2403A52006

MD.Mustafa

Import Required Libraries

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Remove scientific notation globally
pd.set_option('display.float_format', '{:.2f}'.format)
```

Load Dataset

```
# insurance.csv is already uploaded in your Colab
df = pd.read_csv("/content/insurance.csv")

df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Next steps: Generate code with df New interactive sheet

Encode Categorical Columns

```
df_encoded = pd.get_dummies(df, columns=['sex', 'smoker', 'region'], drop_fi

df_encoded.head()
```

	age	bmi	children	charges	sex_male	smoker_yes	region_northwest
0	19	27.900	0	16884.92400	False	True	False
1	18	33.770	1	1725.55230	True	False	False
2	28	33.000	3	4449.46200	True	False	False
3	33	22.705	0	21984.47061	True	False	True
4	32	28.880	0	3866.85520	True	False	True

Next steps: Generate code with df_encoded New interactive sheet

Select Features and Target

```
X = df_encoded[['age', 'bmi', 'children', 'smoker_yes']]
y = df_encoded['charges']
```

Train-Test Split (80/20)

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Apply Feature Scaling (Important for Ridge)

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Train Standard Linear Regression

```
lin_model = LinearRegression()
lin_model.fit(X_train_scaled, y_train)

y_pred_lin = lin_model.predict(X_test_scaled)

mse_lin = mean_squared_error(y_test, y_pred_lin)
r2_lin = r2_score(y_test, y_pred_lin)

print("🔹 Linear Regression Results")
print("MSE:", format(mse_lin, '.2f'))
print("R2 Score:", format(r2_lin, '.4f'))
```

```
🔹 Linear Regression Results
MSE: 33981653.95
R2 Score: 0.7811
```

Train Ridge Regression with Different Alpha Values

```
alphas = [0.1, 1, 10, 100]

ridge_results = []

for alpha in alphas:
    ridge_model = Ridge(alpha=alpha)
    ridge_model.fit(X_train_scaled, y_train)

    y_pred = ridge_model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    ridge_results.append({
        "Alpha": alpha,
        "MSE": round(mse, 2),
        "R2 Score": round(r2, 4)
    })

results_df = pd.DataFrame(ridge_results)

print("\n🔹 Ridge Regression Results")
print(results_df)
```

```
🔹 Ridge Regression Results
   Alpha          MSE  R2 Score
0   0.10  33982227.35      0.78
1   1.00  33987477.22      0.78
2  10.00  34048646.19      0.78
3 100.00  35377800.85      0.77
```

Compare Linear vs Ridge

```
print("Linear Regression MSE:", mse_lin)
print("Linear Regression R2:", r2_lin)

print("\nRidge Results:")
print(results_df)
```

```
Linear Regression MSE: 33981653.95019775
Linear Regression R2: 0.7811147722517887

Ridge Results:
   Alpha           MSE  R2 Score
0    0.1  3.398223e+07  0.781111
```

```
1    1.0  3.398748e+07  0.781077
2   10.0  3.404865e+07  0.780683
3  100.0  3.537780e+07  0.772122
```

Identify Best Alpha

```
best_alpha_row = results_df.loc[results_df['MSE'].idxmin()]

print("\n🏆 Best Alpha (Based on Lowest MSE)")
print(best_alpha_row)
```

```
🏆 Best Alpha (Based on Lowest MSE)
Alpha             0.10
MSE        33982227.35
R2 Score          0.78
Name: 0, dtype: float64
```