EXPERIMENT 2

Shravanya Andhale
D15A-30

AIM: Implement Multi Regression, Lasso, and Ridge Regression on real-world datasets

THEORY:

1. Dataset Source

Dataset provided by Council on Energy, Environment and Water under the India Residential Energy Survey.
Source link:
https://www.kaggle.com/code/raitest/india-residential-energy-survey-ires-2020-eda/input?select=CEEW+-+IRES+Data.csv

2. Dataset Description

The India Residential Energy Survey contains household level information on demographics, dwelling characteristics, economic indicators, and energy usage behaviour across India.

For this experiment, a subset of variables is used.

Predictor variables

- State abbreviation
- Gender of respondent
- Age
- Education of primary income earner
- Number of household members
- Housing type
- Ownership status

These features represent socio economic capacity and infrastructure availability.

Target variable
avg_monthly_bill, a continuous value representing average electricity expenditure.

After preprocessing and removal of missing values, several thousand records are available for modeling.

3. Mathematical Formulation of the Algorithm

Multiple Linear Regression estimates the dependent variable as a linear function of predictors.

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \epsilon$$

Ridge Regression introduces L2 regularization.

$$\min ||y - X\beta||^2 + \alpha||\beta||^2$$

Lasso Regression introduces L1 regularization.

$$\min ||y - X\beta||^2 + \alpha|\beta|$$

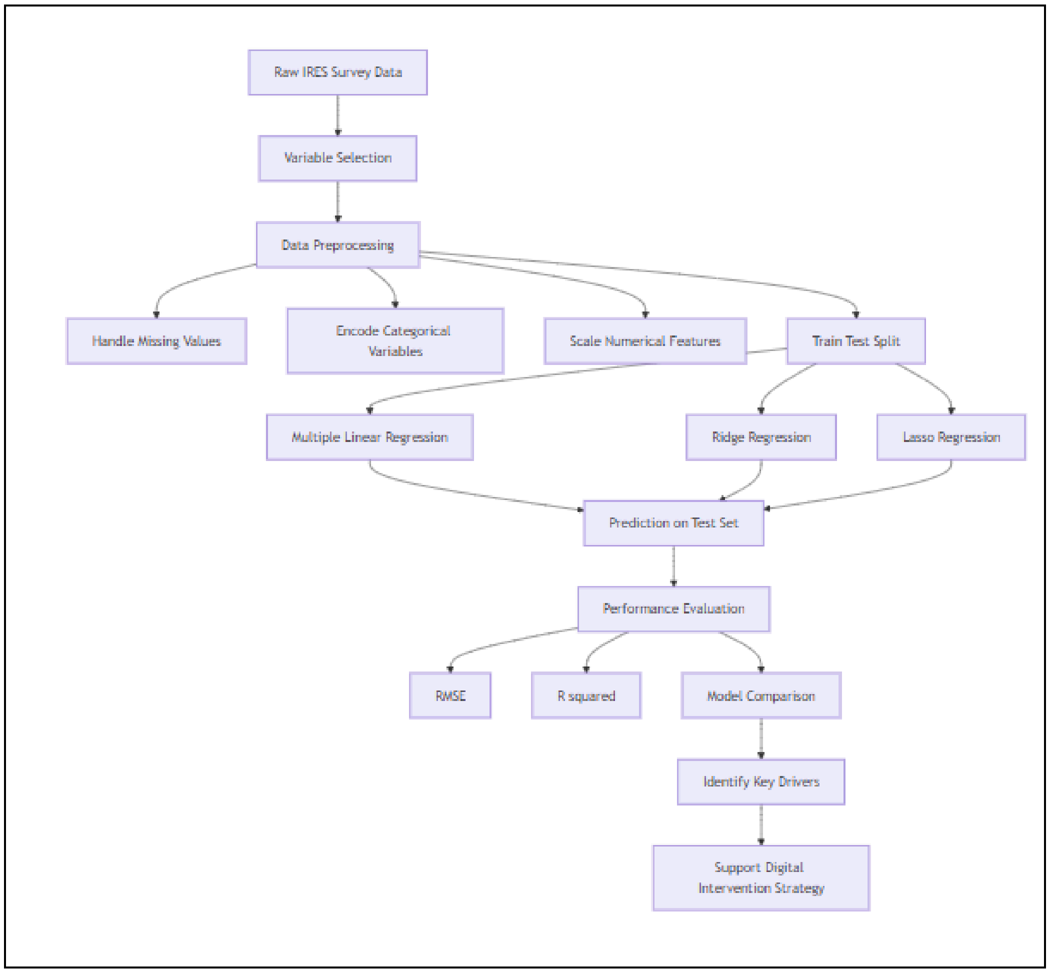
Ridge shrinks coefficients toward zero, while Lasso may force some to become exactly zero.

4. Algorithm Limitations

All models assume linear relationships between variables. They may not capture complex behavioural or nonlinear interactions. Ridge does not perform complete feature elimination. Lasso may arbitrarily select among correlated predictors. Results depend strongly on the quality of survey inputs.

5. Methodology / Workflow

1. Extract relevant predictors and target
2. Remove missing observations
3. Encode categorical variables
4. Standardize numerical features
5. Split into training and testing data
6. Train Multiple Linear Regression as baseline
7. Apply Ridge and Lasso with cross validated hyperparameter tuning
8. Predict on test data
9. Compare error metrics and goodness of fit



6. Performance Analysis

The models produced the following results on the test dataset.

Multiple Linear Regression
RMSE: **601.93**
R squared: **0.2919**

Ridge Regression
Best alpha: **1**
RMSE: **601.88**
R squared: **0.2921**

Lasso Regression
Best alpha: **0.001**

RMSE: **601.93**
R squared: **0.2919**

The performance of all three models is nearly identical. Ridge provides a very small reduction in error, while Lasso behaves almost the same as the baseline model.

This indicates that the predictor set is compact and does not suffer from severe multicollinearity or redundancy. The baseline regression is already stable, and additional regularization does not significantly change predictions.

The R squared value suggests that approximately 29 percent of variation in electricity expenditure is explained by demographic and housing features. Considering the absence of appliance level or climate data, this is a reasonable explanatory strength.

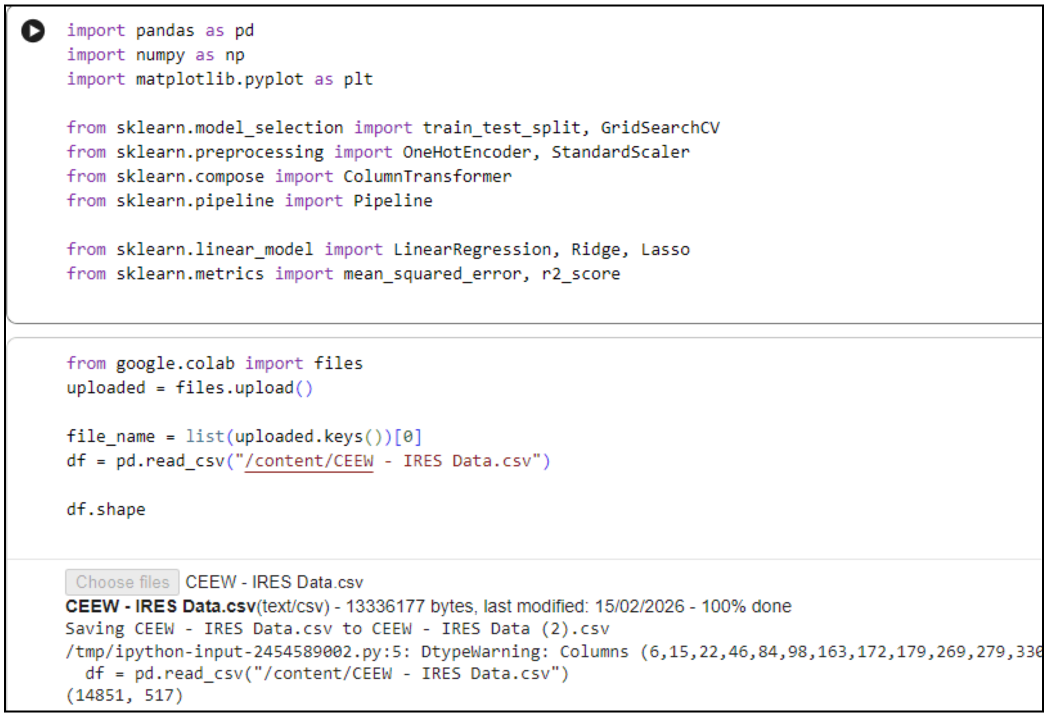
7. Hyperparameter Tuning

Grid search with cross validation was used to determine optimal regularization strength.

For Ridge, alpha equal to 1 provided the best balance between bias and variance.

For Lasso, a very small alpha of 0.001 was selected, indicating minimal need for shrinkage. This further confirms that the model does not contain many weak or noisy predictors.

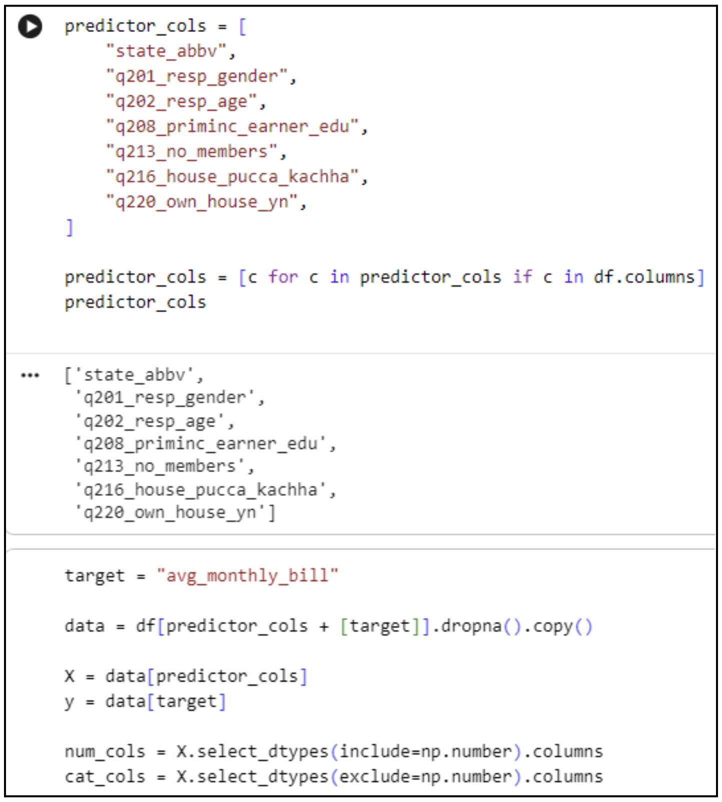
OUTPUT:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
```

```
from google.colab import files
uploaded = files.upload()

file_name = list(uploaded.keys())[0]
df = pd.read_csv("/content/CEEW - IRES Data.csv")

df.shape
```

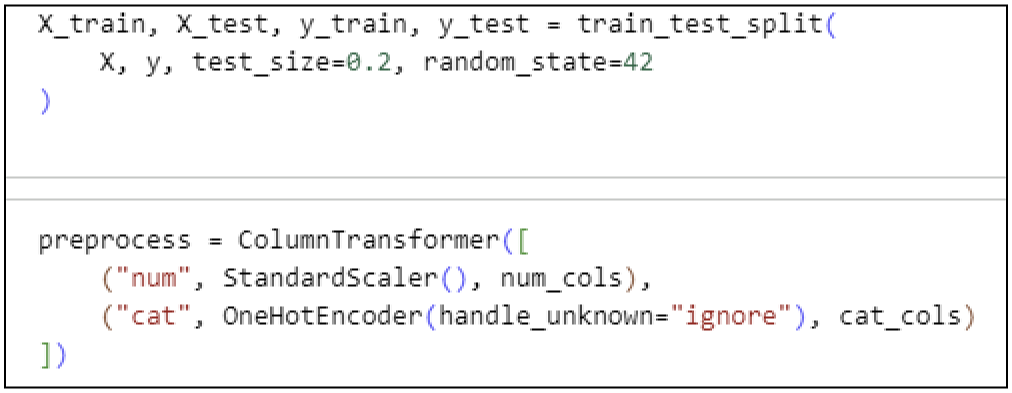
```
Choose files  CEEW - IRES Data.csv
CEEW - IRES Data.csv(text/csv) - 13336177 bytes, last modified: 15/02/2026 - 100% done
Saving CEEW - IRES Data.csv to CEEW - IRES Data (2).csv
/tmp/ipython-input-2454589002.py:5: DtypeWarning: Columns (6,15,22,46,84,98,163,172,179,269,279,33
  df = pd.read_csv("/content/CEEW - IRES Data.csv")
(14851, 517)
```



```
predictor_cols = [
    "state_abbv",
    "q201_resp_gender",
    "q202_resp_age",
    "q208_priminc_earner_edu",
    "q213_no_members",
    "q216_house_pucca_kachha",
    "q220_own_house_yn",
]

predictor_cols = [c for c in predictor_cols if c in df.columns]
predictor_cols
```

```
['state_abbv',
 'q201_resp_gender',
 'q202_resp_age',
 'q208_priminc_earner_edu',
 'q213_no_members',
 'q216_house_pucca_kachha',
 'q220_own_house_yn']
```

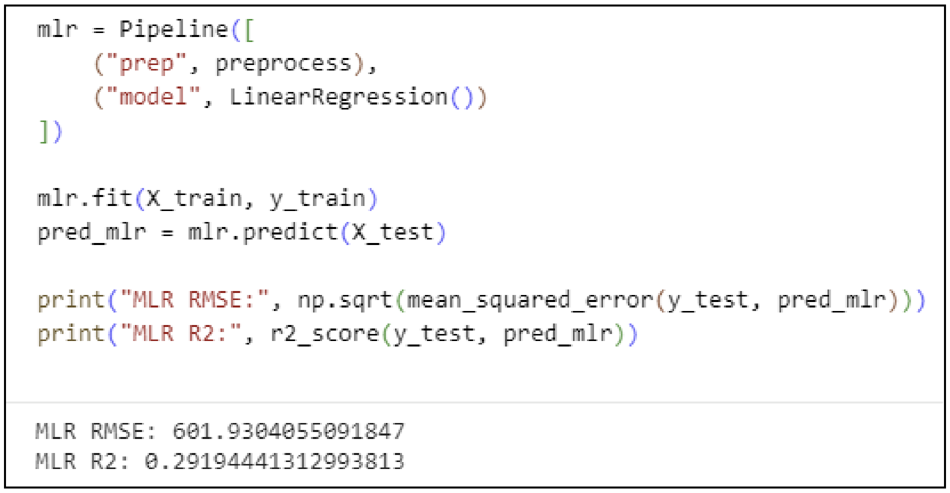
```
target = "avg_monthly_bill"

data = df[predictor_cols + [target]].dropna().copy()

X = data[predictor_cols]
y = data[target]

num_cols = X.select_dtypes(include=np.number).columns
cat_cols = X.select_dtypes(exclude=np.number).columns
```



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

```python
preprocess = ColumnTransformer([
    ("num", StandardScaler(), num_cols),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols)
])
```

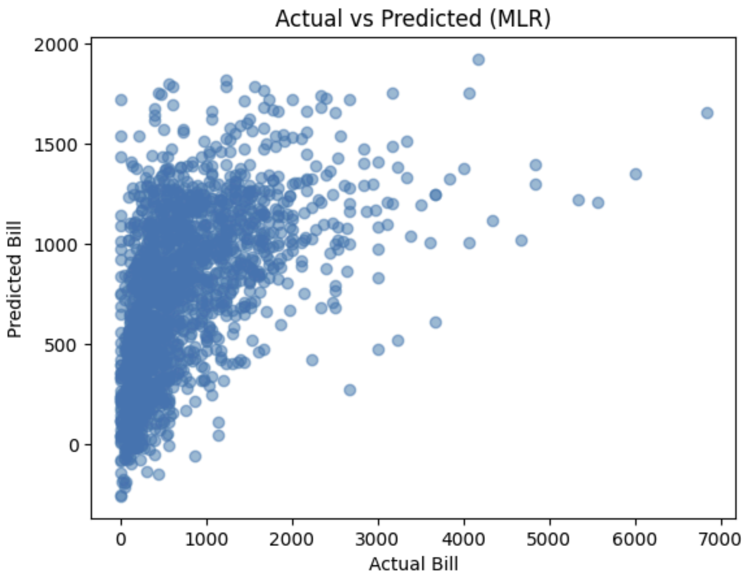
Multiple linear Regression:



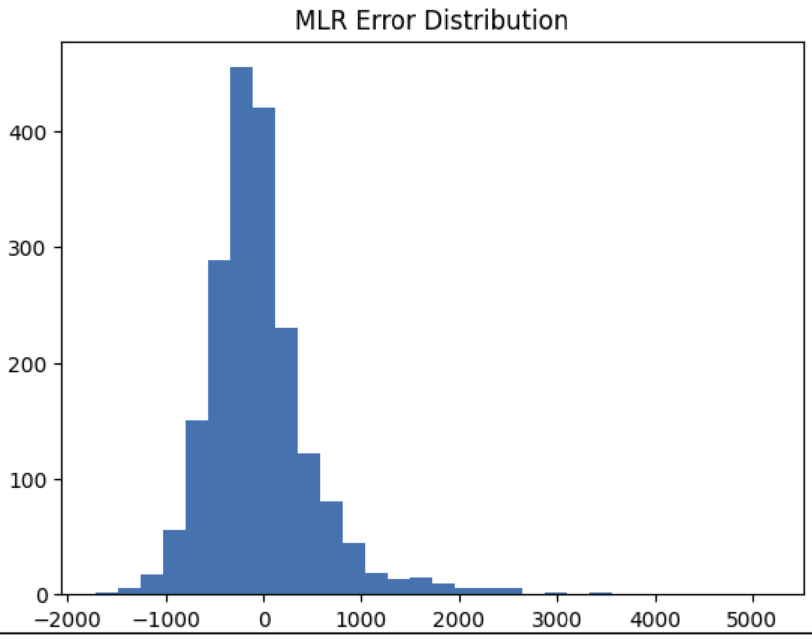
```python
mlr = Pipeline([
    ("prep", preprocess),
    ("model", LinearRegression())
])

mlr.fit(X_train, y_train)
pred_mlr = mlr.predict(X_test)

print("MLR RMSE:", np.sqrt(mean_squared_error(y_test, pred_mlr)))
print("MLR R2:", r2_score(y_test, pred_mlr))
```

```
MLR RMSE: 601.9304055091847
MLR R2: 0.29194441312993813
```

```python
plt.scatter(y_test, pred_mlr, alpha=0.5)
plt.xlabel("Actual Bill")
plt.ylabel("Predicted Bill")
plt.title("Actual vs Predicted (MLR)")
plt.show()
```



```python
errors = y_test - pred_mlr
plt.hist(errors, bins=30)
plt.title("MLR Error Distribution")
plt.show()
```

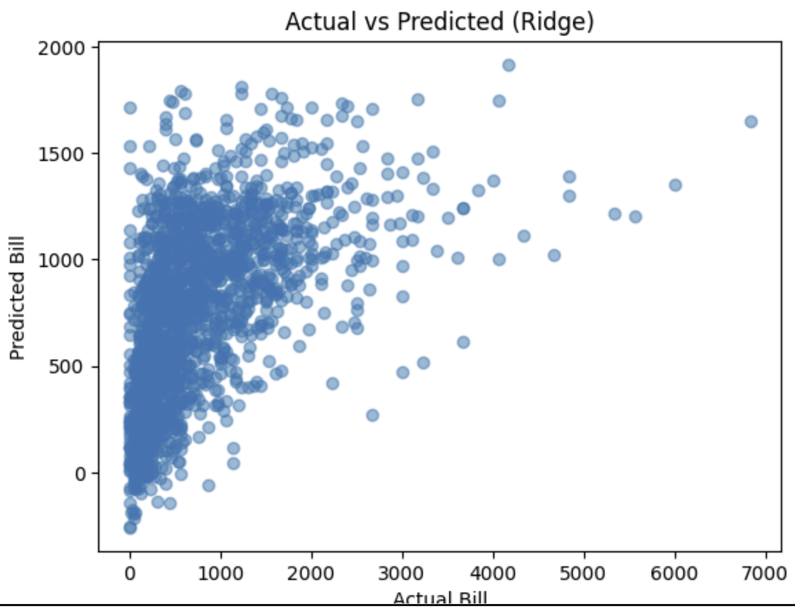


Ridge:

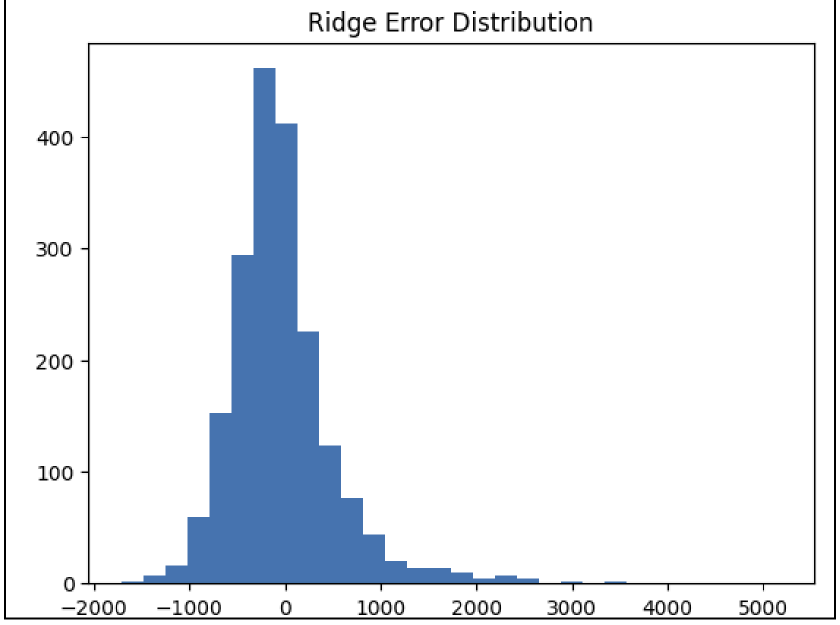
```
ridge = Pipeline([
    ("prep", preprocess),
    ("model", Ridge())
])

param_grid = {"model__alpha": [0.01, 0.1, 1, 10, 100]}

grid_ridge = GridSearchCV(ridge, param_grid, cv=5, scoring="r2")
grid_ridge.fit(X_train, y_train)

best_ridge = grid_ridge.best_estimator_
pred_ridge = best_ridge.predict(X_test)

print("Best alpha:", grid_ridge.best_params_)
print("Ridge RMSE:", np.sqrt(mean_squared_error(y_test, pred_ridge)))
print("Ridge R2:", r2_score(y_test, pred_ridge))
```

```
Best alpha: {'model__alpha': 1}
Ridge RMSE: 601.8757014494711
Ridge R2: 0.2920731049344363
```

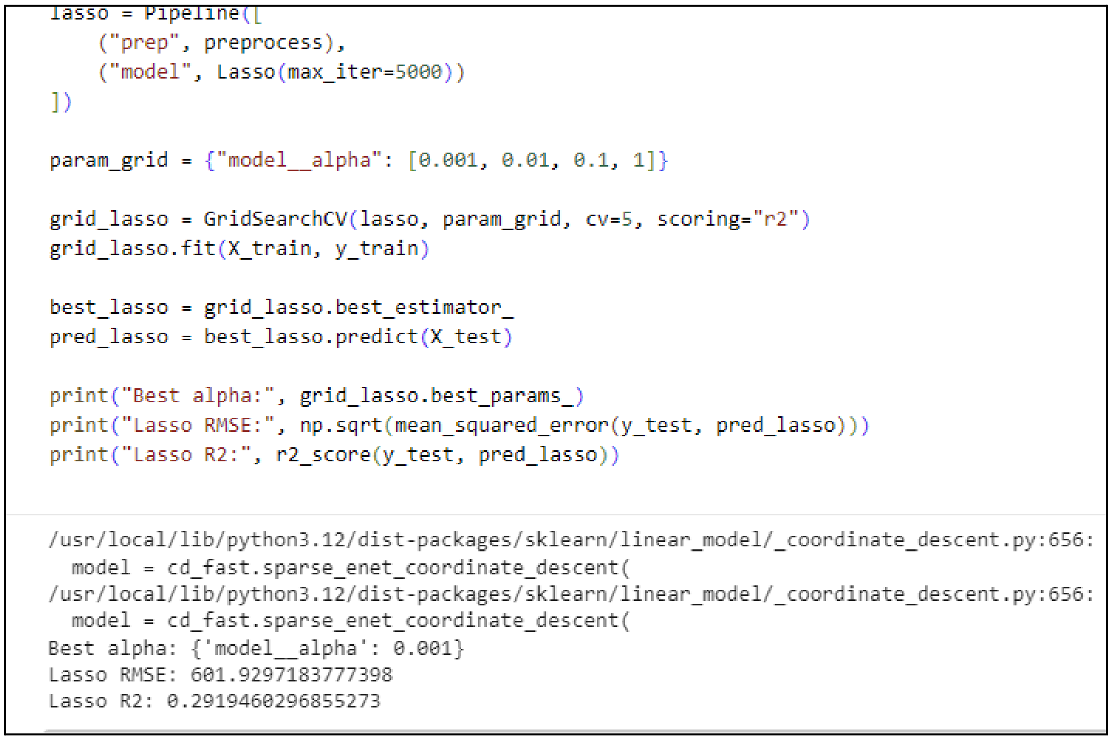
```
plt.scatter(y_test, pred_ridge, alpha=0.5)
plt.xlabel("Actual Bill")
plt.ylabel("Predicted Bill")
plt.title("Actual vs Predicted (Ridge)")
plt.show()
```



```
errors = y_test - pred_ridge
plt.hist(errors, bins=30)
plt.title("Ridge Error Distribution")
plt.show()
```



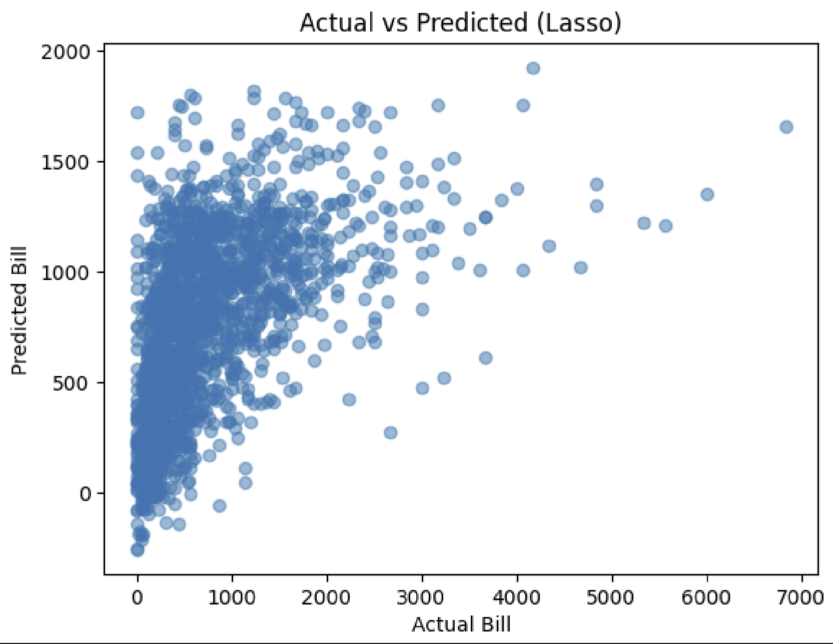
Lasso:



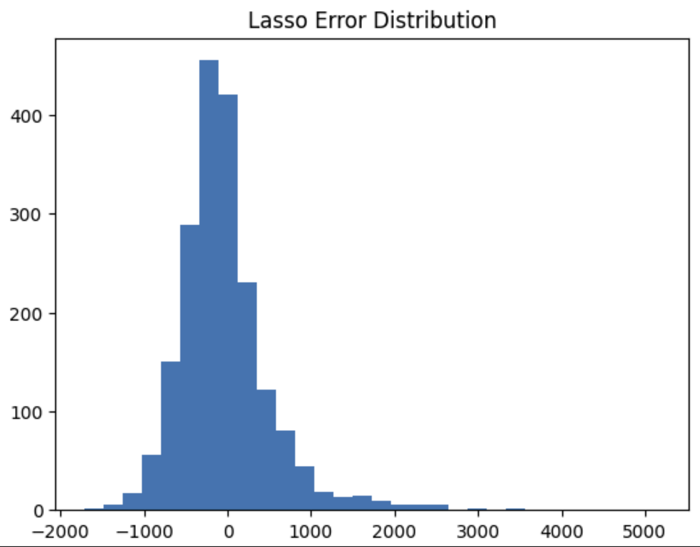
```
lasso = Pipeline([
    ("prep", preprocess),
    ("model", Lasso(max_iter=5000))
])

param_grid = {"model__alpha": [0.001, 0.01, 0.1, 1]}

grid_lasso = GridSearchCV(lasso, param_grid, cv=5, scoring="r2")
grid_lasso.fit(X_train, y_train)

best_lasso = grid_lasso.best_estimator_
pred_lasso = best_lasso.predict(X_test)

print("Best alpha:", grid_lasso.best_params_)
print("Lasso RMSE:", np.sqrt(mean_squared_error(y_test, pred_lasso)))
print("Lasso R2:", r2_score(y_test, pred_lasso))
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_coordinate_descent.py:656:
  model = cd_fast.sparse_enet_coordinate_descent(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_coordinate_descent.py:656:
  model = cd_fast.sparse_enet_coordinate_descent(
Best alpha: {'model__alpha': 0.001}
Lasso RMSE: 601.9297183777398
Lasso R2: 0.2919460296855273
```

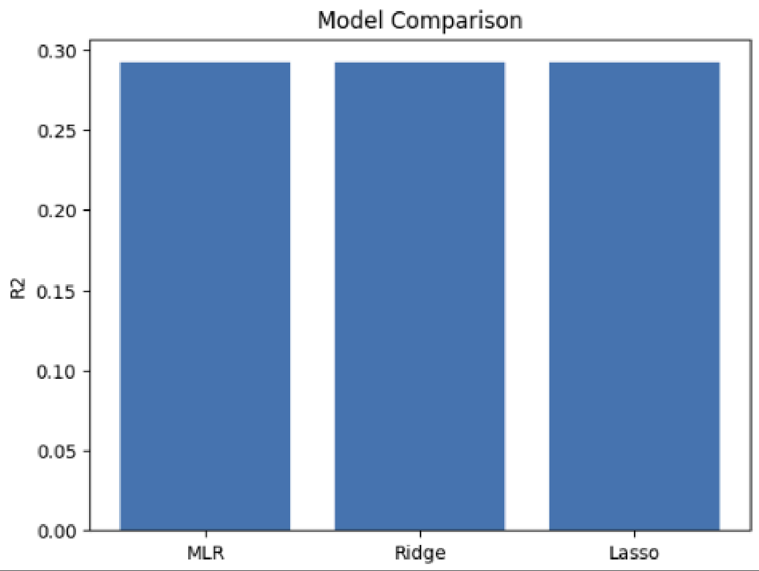
```
plt.scatter(y_test, pred_lasso, alpha=0.5)
plt.xlabel("Actual Bill")
plt.ylabel("Predicted Bill")
plt.title("Actual vs Predicted (Lasso)")
plt.show()
```

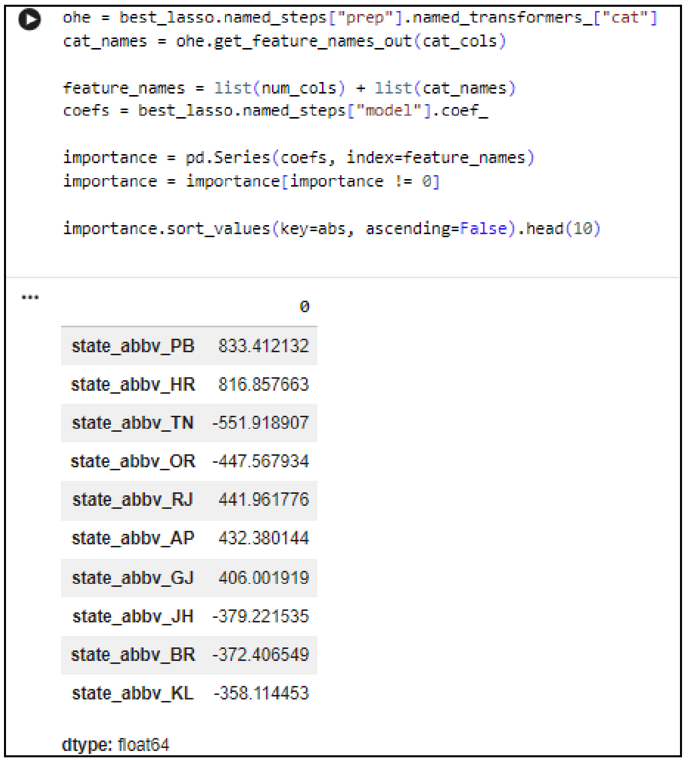


```
errors = y_test - pred_lasso
plt.hist(errors, bins=30)
plt.title("Lasso Error Distribution")
plt.show()
```



```python
models = ["MLR", "Ridge", "Lasso"]
r2_scores = [
    r2_score(y_test, pred_mlr),
    r2_score(y_test, pred_ridge),
    r2_score(y_test, pred_lasso)
]

plt.bar(models, r2_scores)
plt.ylabel("R2")
plt.title("Model Comparison")
plt.show()
```





```python
ohe = best_lasso.named_steps["prep"].named_transformers_["cat"]
cat_names = ohe.get_feature_names_out(cat_cols)

feature_names = list(num_cols) + list(cat_names)
coefs = best_lasso.named_steps["model"].coef_

importance = pd.Series(coefs, index=feature_names)
importance = importance[importance != 0]

importance.sort_values(key=abs, ascending=False).head(10)
```

```
                        0
state_abbv_PB   833.412132
state_abbv_HR   816.857663
state_abbv_TN  -551.918907
state_abbv_OR  -447.567934
state_abbv_RJ   441.961776
state_abbv_AP   432.380144
state_abbv_GJ   406.001919
state_abbv_JH  -379.221535
state_abbv_BR  -372.406549
state_abbv_KL  -358.114453

dtype: float64
```

CONCLUSION: Multiple Linear, Ridge, and Lasso Regression were successfully implemented on real world survey data. The similarity in performance across models demonstrates robustness of the predictor set and absence of excessive noise. Regularization validated that demographic and housing characteristics provide consistent explanatory power. These findings support development of simple, interpretable, and scalable digital systems for identifying households that may benefit from energy efficiency guidance.