In [1]:
```
from sklearn.datasets import fetch_california_housing
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import statsmodels.api as sm
import nbformat as nbf
import sympy as sp
from sympy import symbols, Matrix, diff
import random
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures

from sklearn.preprocessing import StandardScaler
from matplotlib.animation import FuncAnimation
from sklearn.metrics import mean_squared_error, r2_score
import pickle
from prettytable import PrettyTable
```

In [2]:
```
df = pd.read_pickle("preprocessed_df.pkl")  # or read_csv
print(df.info(memory_usage="deep"))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   HouseAge                 20640 non-null  float64
 1   Medinc_log               20640 non-null  float64
 2   AveRooms_boxcox          20640 non-null  float64
 3   AveBedrms_boxcox         20640 non-null  float64
 4   Population_log           20640 non-null  float64
 5   AveOccup_boxcox          20640 non-null  float64
 6   spatial_proximity_score  20640 non-null  float64
 7   MedHouseVal              20640 non-null  float64
dtypes: float64(8)
memory usage: 1.3 MB
None
```

Task - 1 : Formulating this as LASSO, and specify the Objective Function and Convex but not Differentiable

Formulating this as Lasso :

We need to predict the target from features X :

We want to predict a target y from features X.

Model:

$$\hat{y} = Xw$$

Goal:

$$\min_w \ \|y - Xw\|_2^2$$

This is just **least squares.**

Now add the penalty term, such that it reduces the weights of the Irrelevant features to 0

Now, LASSO formulation is simply:

$$\min_w \ \underbrace{\|y - Xw\|_2^2}_{\text{fit data}} + \underbrace{\lambda \|w\|_1}_{\text{sparsity}}$$

The Final **Objective Function** Would be :

$$\min_w \ F(w) = \|y - Xw\|_2^2 + \lambda \sum_i |w_i|$$

Why This Is a Convex Problem

- Squared loss is a **smooth convex** curve (bowl shape).
- L1 norm is a **convex** V-shape.
- Sum of two convex functions is **always convex.**

So the whole LASSO problem is **convex.**

Why It Is Non-Differentiable

- L1 norm has a **sharp corner at 0.**
- At this point, derivative does not exist.
- That's why LASSO is **convex but not differentiable**.

Why L2 (Ridge) Is Not Enough

- L2 penalty shrinks all weights smoothly.

- It **never makes coefficients exactly zero**.
- No feature selection happens.
- Still sensitive to outliers.

Why L1 Is Used

- L1 can make weights **exactly zero**.
- Gives **feature selection** automatically.
- Keeps model simple.
- Works well when many features exist.

Solutions For this Problem :

We use:

- **Subgradient methods**
- **Proximal gradient methods (Soft thresholding)**

These methods handle the sharp corner of L1.

Task - 2 : Derive soft thresholding used in ISTA with update step

Lasso defined as :

$$\min_{w} \; F(w) = \|y - Xw\|_2^2 + \lambda \sum_i |w_i|$$

$$\min_{w \in \mathbb{R}^d} \; F(w) \; = \; f(w) + g(w)$$

$$f(w) = \frac{1}{2n}\|y - Xw\|_2^2 \quad \text{(smooth)} \qquad g(w) = \lambda \|w\|_1 \quad \text{(non-smooth)}$$

Goal : Solve for **(w*)**

- f has a gradient ∇f (we can do gradient steps).
- g is convex but non-differentiable at 0 (we handle via *proximal operator*).
- This splitting lets us alternate a gradient step on f and a proximal step on g.

Proximal Operator is Defined as follows :

For any convex function g and scalar $\tau > 0$,

$$\text{prox}_{\tau g}(v) = \arg\min_{w} \frac{1}{2}\|w - v\|_2^2 + \tau g(w).$$

Idea of ISTA(Iterative Shrinkage thresholding Algorithm):

Gradient step on smooth part f:

$$v = w^k - \alpha \nabla f(w^k)$$

Prox step for g:

$$w^{k+1} = \text{prox}_{\alpha g}(v)$$

With $f(w) = \frac{1}{2n}\|y - Xw\|^2$,

$$\nabla f(w) = -\frac{1}{n}X^\top(y - Xw) = \frac{1}{n}X^\top(Xw - y).$$

So gradient step:

$$v = w^k - \alpha \cdot \frac{1}{n}X^\top(Xw^k - y).$$

So, We need the proximity for L1, so we reduce it to the 1D as follows :

$$\min_{w_j} \frac{1}{2}(w_j - v_j)^2 + \alpha\lambda|w_j|.$$

So, the Solution for this would be :

Consider scalar $q(w) = \frac{1}{2}(w - v)^2 + \alpha\lambda|w|$.

- **Case A:** $w > 0$

  $|w| = w$. Differentiate:

  $$q'(w) = (w - v) + \alpha\lambda.$$

  Set $q'(w) = 0 \Rightarrow w = v - \alpha\lambda$. Valid if $v - \alpha\lambda > 0$, i.e. $v > \alpha\lambda$.
- **Case B:** $w < 0$

  $|w| = -w$. Differentiate:

  $$q'(w) = (w - v) - \alpha\lambda.$$

  Set to zero $\Rightarrow w = v + \alpha\lambda$. Valid if $v + \alpha\lambda < 0$, i.e. $v < -\alpha\lambda$.
- **Case C:** $w = 0$

  $w = 0$ optimal when $|v| \leq \alpha\lambda$ (subgradient condition).

Final Thresholding Proximity Formula Will be :

$$w^\star = S_{\alpha\lambda}(v) = \begin{cases} v - \alpha\lambda & v > \alpha\lambda, \\ 0 & |v| \leq \alpha\lambda, \\ v + \alpha\lambda & v < -\alpha\lambda. \end{cases}$$

And the Compact Form would be :

$$S_t(v) = \operatorname{sign}(v)\max(|v| - t, 0), \qquad t = \alpha\lambda.$$

ISTA Update Would be

$$v = w^k - \alpha\nabla f(w^k) = w^k - \alpha \cdot \frac{1}{n}X^\top(Xw^k - y).$$

$$w^{k+1} = S_{\alpha\lambda}\left(w^k - \alpha \cdot \frac{1}{n} X^\top (Xw^k - y)\right)$$

Safe Step Size Rule Would be

Let L be Lipschitz constant of ∇f. For $f = \frac{1}{2n}\|y - Xw\|^2$,

$$L = \frac{1}{n}\lambda_{\max}(X^\top X).$$

Choose $\alpha \in (0, 1/L]$. Common choice: $\alpha = 1/L$. Then ISTA converges at rate $O(1/k)$. (FISTA accelerates to $O(1/k^2)$.)

$$\text{SoftThresh}(z_i, \tau) = \text{sign}(z_i)\max(|z_i| - \tau, 0)$$

Task - 3 : Implement the Lasso with ISTA with λ = [0.01, 0.1, 1]

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   HouseAge                20640 non-null  float64
 1   Medinc_log              20640 non-null  float64
 2   AveRooms_boxcox         20640 non-null  float64
 3   AveBedrms_boxcox        20640 non-null  float64
 4   Population_log          20640 non-null  float64
 5   AveOccup_boxcox         20640 non-null  float64
 6   spatial_proximity_score 20640 non-null  float64
 7   MedHouseVal             20640 non-null  float64
dtypes: float64(8)
memory usage: 1.3 MB
```

In [12]:
```
df = df.copy()

X = df.drop(columns=['MedHouseVal']).values
y = df['MedHouseVal'].values.reshape(-1, 1)

# Normalize features
X_mean = X.mean(axis=0)
X_std = X.std(axis=0) + 1e-8
X = (X - X_mean) / X_std

n, d = X.shape

# Gradient Descent for Ridge Regression
def ridge_gradient_descent(X, y, lam, lr=0.01, iters=500):
    n, d = X.shape
    w = np.zeros((d,1))
```

```
    for _ in range(iters):
        grad = (1/n) * (X.T @ (X@w - y)) + lam * w
        w -= lr * grad
    return w

# Soft Thresholding

def soft_thresholding(z, alpha):
    return np.sign(z) * np.maximum(np.abs(z) - alpha, 0)

# ISTA Updation

def lasso_ista(X, y, lam, lr=0.01, iters=1000):
    n, d = X.shape
    w = np.zeros((d,1))

    for _ in range(iters):
        grad = (1/n) * (X.T @ (X@w - y))
        w = soft_thresholding(w - lr*grad, lr*lam)
    return w


lambdas = [0.01, 0.1, 1, 0.005, 0.05, 0.5, 2e-2, 2e-1, 5e-2, 5e-1, 2e-3, 2e-4, 5e-3

import pandas as pd

results = {}

for lam in lambdas:
    print(f"\n========== λ = {lam} ==========")

    w_ridge = ridge_gradient_descent(X, y, lam)
    w_lasso = lasso_ista(X, y, lam)

    results[lam] = {"ridge": w_ridge, "lasso": w_lasso}

    table = pd.DataFrame({
        "Feature": df.drop(columns=['MedHouseVal']).columns,
        "Ridge Weight": w_ridge.flatten(),
        "LASSO Weight": w_lasso.flatten()
    })

    print(table.to_string(index=False))
```

```
========== λ = 0.01 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.113594      0.111920
             Medinc_log      0.758464      0.829227
        AveRooms_boxcox     -0.034653     -0.109466
       AveBedrms_boxcox      0.038826      0.075311
         Population_log      0.028671      0.004826
        AveOccup_boxcox     -0.290120     -0.268938
spatial_proximity_score      0.288159      0.256074

========== λ = 0.1 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.104358      0.024214
             Medinc_log      0.686386      0.660326
        AveRooms_boxcox     -0.004207     -0.000000
       AveBedrms_boxcox      0.021063      0.000000
         Population_log      0.024802     -0.000000
        AveOccup_boxcox     -0.269410     -0.194263
spatial_proximity_score      0.274062      0.226851

========== λ = 1 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.058713           0.0
             Medinc_log      0.361491           0.0
        AveRooms_boxcox      0.069598           0.0
       AveBedrms_boxcox     -0.024338          -0.0
         Population_log      0.005728          -0.0
        AveOccup_boxcox     -0.153757          -0.0
spatial_proximity_score      0.172098           0.0

========== λ = 0.005 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.114158      0.119145
             Medinc_log      0.762920      0.843817
        AveRooms_boxcox     -0.036653     -0.123005
       AveBedrms_boxcox      0.039990      0.086597
         Population_log      0.028903      0.010981
        AveOccup_boxcox     -0.291352     -0.273240
spatial_proximity_score      0.288952      0.255212

========== λ = 0.05 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.109285      0.068896
             Medinc_log      0.724613      0.713657
        AveRooms_boxcox     -0.019890     -0.000000
       AveBedrms_boxcox      0.030226      0.000000
         Population_log      0.026883      0.000000
        AveOccup_boxcox     -0.280580     -0.242108
spatial_proximity_score      0.281843      0.258058

========== λ = 0.5 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.077191      0.000000
             Medinc_log      0.486109      0.273021
        AveRooms_boxcox      0.056190      0.000000
       AveBedrms_boxcox     -0.014958     -0.000000
```

```
         Population_log      0.013023     -0.000000
         AveOccup_boxcox     -0.202798     -0.000000
spatial_proximity_score      0.219646      0.000000

========== λ = 0.02 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.112484      0.099044
             Medinc_log      0.749706      0.798062
         AveRooms_boxcox     -0.030759     -0.079704
        AveBedrms_boxcox      0.036560      0.051558
          Population_log      0.028213      0.000000
         AveOccup_boxcox     -0.287682     -0.261978
spatial_proximity_score      0.286574      0.257867

========== λ = 0.2 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.095794      0.000000
             Medinc_log      0.621217      0.563829
         AveRooms_boxcox      0.019810     -0.000000
        AveBedrms_boxcox      0.006948      0.000000
          Population_log      0.021109     -0.000000
         AveOccup_boxcox     -0.249309     -0.096857
spatial_proximity_score      0.259041      0.139529

========== λ = 0.05 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.109285      0.068896
             Medinc_log      0.724613      0.713657
         AveRooms_boxcox     -0.019890     -0.000000
        AveBedrms_boxcox      0.030226      0.000000
          Population_log      0.026883      0.000000
         AveOccup_boxcox     -0.280580     -0.242108
spatial_proximity_score      0.281843      0.258058

========== λ = 0.5 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.077191      0.000000
             Medinc_log      0.486109      0.273021
         AveRooms_boxcox      0.056190      0.000000
        AveBedrms_boxcox     -0.014958     -0.000000
          Population_log      0.013023     -0.000000
         AveOccup_boxcox     -0.202798     -0.000000
spatial_proximity_score      0.219646      0.000000

========== λ = 0.002 ==========
                Feature  Ridge Weight  LASSO Weight
              HouseAge      0.114499      0.123445
             Medinc_log      0.765619      0.852268
         AveRooms_boxcox     -0.037871     -0.130798
        AveBedrms_boxcox      0.040699      0.093177
          Population_log      0.029042      0.014728
         AveOccup_boxcox     -0.292096     -0.275885
spatial_proximity_score      0.289429      0.254818

========== λ = 0.0002 ==========
                Feature  Ridge Weight  LASSO Weight
```

```
                HouseAge      0.114705      0.126010
              Medinc_log      0.767248      0.857219
        AveRooms_boxcox     -0.038609     -0.135342
       AveBedrms_boxcox      0.041128      0.097049
         Population_log      0.029127      0.016996
        AveOccup_boxcox     -0.292544     -0.277498
spatial_proximity_score      0.289715      0.254630

========== λ = 0.005 ==========
                 Feature  Ridge Weight  LASSO Weight
                HouseAge      0.114158      0.119145
              Medinc_log      0.762920      0.843817
        AveRooms_boxcox     -0.036653     -0.123005
       AveBedrms_boxcox      0.039990      0.086597
         Population_log      0.028903      0.010981
        AveOccup_boxcox     -0.291352     -0.273240
spatial_proximity_score      0.288952      0.255212

========== λ = 0.0005 ==========
                 Feature  Ridge Weight  LASSO Weight
                HouseAge      0.114671      0.125583
              Medinc_log      0.766976      0.856401
        AveRooms_boxcox     -0.038485     -0.134593
       AveBedrms_boxcox      0.041056      0.096408
         Population_log      0.029113      0.016617
        AveOccup_boxcox     -0.292469     -0.277228
spatial_proximity_score      0.289667      0.254658
```

Task - 4 : Convergence of Training plot, MSE and R2, Sparsity Ratio

In [5]:
```
# For λ = 0.1 Convergence plot for LASSO training objective Vs iterations using plo

lam = 0.1
n, d = X.shape
w = np.zeros((d,1))
iters = 1000
objectives = []
for it in range(iters):
    grad = (1/n) * (X.T @ (X@w - y))
    w = soft_thresholding(w - 0.01*grad, 0.01*lam)

    residual = y - X @ w
    lasso_obj = (1/(2*n)) * np.sum(residual**2) + lam * np.sum(np.abs(w))
    objectives.append(lasso_obj)

fig = px.line(x=list(range(iters)), y=objectives, labels={'x':'Iterations', 'y':'LA
              title='LASSO Training Objective vs Iterations (λ=0.1)')
fig.show()
```

In [6]:
```
# For λ = 0.01 Convergence plot for LASSO training objective Vs iterations using pl

lam = 0.01
n, d = X.shape
```

```
w = np.zeros((d,1))
iters = 1000
objectives = []
for it in range(iters):
    grad = (1/n) * (X.T @ (X@w - y))
    w = soft_thresholding(w - 0.01*grad, 0.01*lam)

    residual = y - X @ w
    lasso_obj = (1/(2*n)) * np.sum(residual**2) + lam * np.sum(np.abs(w))
    objectives.append(lasso_obj)
fig = px.line(x=list(range(iters)), y=objectives, labels={'x':'Iterations', 'y':'LA
              title='LASSO Training Objective vs Iterations (λ=0.01)')
fig.show()
```

In [7]:
```
# For λ = 1 Convergence plot for LASSO training objective Vs iterations using plotl

lam = 1
n, d = X.shape
w = np.zeros((d,1))
iters = 1000
objectives = []
for it in range(iters):
    grad = (1/n) * (X.T @ (X@w - y))
    w = soft_thresholding(w - 0.01*grad, 0.01*lam)

    residual = y - X @ w
    lasso_obj = (1/(2*n)) * np.sum(residual**2) + lam * np.sum(np.abs(w))
    objectives.append(lasso_obj)
fig = px.line(x=list(range(iters)), y=objectives, labels={'x':'Iterations', 'y':'LA
              title='LASSO Training Objective vs Iterations (λ=1)')
fig.show()
```

In [13]:
```
# Report MSE, R2 and Sparsity Ratio for different λ values in list for LASSO in Pre

table = PrettyTable()
table.field_names = ["λ", "MSE", "R2 Score", "Sparsity Ratio"]
for lam in lambdas:
    w_lasso = results[lam]["lasso"]
    y_pred = X @ w_lasso

    mse = mean_squared_error(y, y_pred)
    r2 = r2_score(y, y_pred)
    sparsity_ratio = np.sum(w_lasso == 0) / len(w_lasso)

    table.add_row([lam, round(mse, 4), round(r2, 4), round(sparsity_ratio, 4)])
print(table)
```

λ	MSE	R2 Score	Sparsity Ratio
0.01	4.7935	-2.5999	0.0
0.1	4.8432	-2.6372	0.4286
1	5.6105	-3.2135	1.0
0.005	4.792	-2.5988	0.0
0.05	4.8166	-2.6173	0.4286
0.5	5.2629	-2.9525	0.8571
0.02	4.7977	-2.6031	0.1429
0.2	4.9335	-2.7051	0.5714
0.05	4.8166	-2.6173	0.4286
0.5	5.2629	-2.9525	0.8571
0.002	4.7914	-2.5983	0.0
0.0002	4.7911	-2.5981	0.0
0.005	4.792	-2.5988	0.0
0.0005	4.7911	-2.5981	0.0

Very bad Move, as R score is -ve which means underfitting, i.e., the data might not be sufficient, so we are introducing the new features

In [9]:
```
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)   # X is your normalized original data

print("Original shape:", X.shape)
print("Polynomial shape:", X_poly.shape)
```

```
Original shape: (20640, 7)
Polynomial shape: (20640, 35)
```

In [10]:
```
X_mean = X_poly.mean(axis=0)
X_std  = X_poly.std(axis=0) + 1e-8
X_poly = (X_poly - X_mean) / X_std
```

In [11]:
```
w_lasso   = lasso_ista(X_poly, y, lam)

def predict(X, w):
    return X @ w


def mse(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

def r2_score_custom(y_true, y_pred):
    ss_res = np.sum((y_true - y_pred) ** 2)
    ss_tot = np.sum((y_true - np.mean(y_true)) ** 2)
    return 1 - (ss_res / ss_tot)

y_pred_poly = predict(X_poly, w_lasso)
mse_poly = mse(y, y_pred_poly)
r2_poly = r2_score_custom(y, y_pred_poly)
print(f"Polynomial Features LASSO Regression - MSE: {mse_poly:.4f}, R2 Score: {r2_p
```

```
Polynomial Features LASSO Regression - MSE: 5.6105, R2 Score: -3.2135
```

Final Report

Model Setup

- Model: LASSO Regression
- Feature Engineering: Polynomial Features (degree = 2)
- Objective: Evaluate whether polynomial expansion improves predictive performance.

Model Metrics

Metric	Value
MSE	**5.6105**
R^2 Score	**–3.2135**

Interpretation

- R^2 is strongly negative → model performs worse than predicting the mean.
- MSE increased → no improvement after polynomial feature expansion.
- LASSO zeroed out most polynomial terms → high sparsity → strong underfitting.

❌ This model is not suitable for this dataset.

Polynomial Features + LASSO does **not** improve accuracy and makes the model perform significantly worse.