Análisis y Reporte sobre el desempeño del modelo

1. Implementación elegida: uso de framework, sklearn (linear_model)

   a. Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation).

```
[17] # Score the model
     reg.score(x_test, y_test)

     0.8513618855525324

[18] reg.score(x_train, y_train)

     0.8561208934839917

     scores = cross_val_score(reg, x_train, y_train, cv=5, scoring='r2')
     print("Mean score of %0.2f with a standard deviation of %0.2f" % (scores.mean(), scores.std()))

     Mean score of 0.86 with a standard deviation of 0.01
```

   b. Diagnóstico y explicación el grado de bias y varianza: bajo medio alto

```
avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(reg, x_train, y_train, x_test, y_test, loss='mse', num_rounds=50, random_seed=20)

# summary of the results
print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

Al tratarse de una regresión lineal el bias es alto debido a que el modelo es poco flexible al ajuste de los datos, por otro lado, tiene una varianza baja (cambiar los datos de entrenamiento produce cambios pequeños en la estimación).

   c. Diagnóstico y explicación el nivel de ajuste del modelo: underfitt fitt overfitt

```
[76] # Model Information
     print("Intercept:", reg.intercept_)
     print("Coeficiente:", list(zip(x_test.columns, reg.coef_.flatten(), )))
     print("Coeficiente de determinación R^2:", reg.score(x_test, y_test))

     Intercept: [-351.02587529]
     Coeficiente: [('height', 7.7225058871406205)]
     Coeficiente de determinación R^2: 0.8513618855525324

     # Model test error
     predicciones = reg.predict(X = x_test)
     rmse = mean_squared_error(y_true  = y_test,y_pred  = predicciones,squared = False)
     print(f"El error (rmse) de test es: {rmse}")

     El error (rmse) de test es: 12.372434423109533
```

2. Técnicas de regularización implementadas: Lasso y Ridge

a. Resultados:

```
# Lasso model regularization
lasso_reg = Lasso(alpha=0.3)
lasso_reg.fit(x_train, y_train)
```

```
Lasso(alpha=0.3)
```

```
[68] lasso_reg.score(x_test, y_test)
```

```
0.8513712187225047
```

```
[69] lasso_reg.score(x_train, y_train)
```

```
0.8561149964826482
```

```
[70] # Ridge model regularization
ridge_reg = Ridge(alpha=0.3)
ridge_reg.fit(x_train, y_train)
```

```
Ridge(alpha=0.3)
```

```
[71] ridge_reg.score(x_test, y_test)
```

```
0.8513619002560128
```

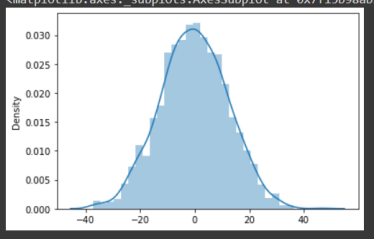
```
[72] ridge_reg.score(x_train, y_train)
```

```
0.8561208934784967
```

```
[73] # Lasso and ridge predictions
prediction_lasso=lasso_reg.predict(x_test)
prediction_ridge=ridge_reg.predict(x_test)
```

```
[74] prediction_lasso=prediction_lasso.reshape(2000,1)
sns.distplot(y_test-prediction_lasso)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distri
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f15b98ab
```




```
sns.distplot(y_test-prediction_ridge)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f15b9774550>
```