Module 5 exercises: Regression

Dataset characteristics

California Housing dataset characteristics:

- Number of Instances: 20640
- Number of Attributes: 8 numeric, predictive attributes and the target
- Attribute Information:
    - MedInc median income in block group
    - HouseAge median house age in block group
    - AveRooms average number of rooms per household
    - AveBedrms average number of bedrooms per household
    - Population block group population
    - AveOccup average number of household members
    - Latitude block group latitude
    - Longitude block group longitude

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, these columns may take surpinsingly large values for block groups with few households and many empty houses, such as vacation resorts.

The goal of this notebook is to build a Linear Regression Model on the California Housing Dataset. We will be trying to predict the MedianPrice per house in each block. Denoted as the target column. This price is in $100,000s

We have two datasets available:

- `housing_data.csv` is the original dataset
- `housing_data_cleaned_STD.csv` is the same dataset but having had 114 rows removed with missing AveRooms. It has also had all points above or below 3 standard deviations removed.

Exercise 1 – Setup

1. Import the following packages:
    i. Pandas
    ii. matplotlib
    iii. numpy
    iv. seaborn
2. Load the following data file into a `DataFrame`:
    i. `housing_data.csv`

Exercise 2 – EDA

Familiarise yourself with the dataset. Seek to understand:

- the types of data.
- distributions of features.
- obvious patterns.
- errors, nulls, outliers.

Exercise 3 – Assumptions

1. Select the features you would like to use to build a regression model.
2. Determine whether any assumtions of linear regression would be violated by using those features.
3. How might you mitigate these problems?

Exercise 4 – Feature engineering

1. Inspect the data. Are there any features that could be created by combining existing ones together?
2. Try to construct a novel feature from existing ones. Bedroom ratio may be predictive, so we choose to incorporate it.

Exercise 5 – Preprocessing

1. So we can evaluate our model, we split our data into a training and testing dataset. Why do we do this?

2. Use scikit-learn's `train_test_split` function, found in the `preprocessing` module, to do this. Set `random_state` to 42 and `test_size` to 0.30 You should end up with 4 sets of data after doing so: `X_train`, `X_test`, `y_train`, `y_test`.
3. We then make the decicision to scale the data. Why might we choose to scale the data in a regression modelling project?
4. Create, fit, and apply two StandardScalers, also found in preprocessing, to `X_train` and `y_train`.
5. Apply the one used for X to `X_test` and the one usef for y to `y_test`. Why might we do this?

Exercise 6 – Building your first model

1. After preprocessing the dataset, we can fit our regression model. Import the `LinearRegression` model from scikit-learn's `linear_model` module.
2. Create a `LinearRegression` model, setting `fit_intercept=False`. Then use the `fit` method to train the model on `X_train` and `y_train`.
3. Generate predictions from the model for `X_test` using the `predict` method, storing them in the variable `y_pred`.

Exercise 7 - Plotting your model

Run the below code to plot your model against the training data. How well do you think it fits?

Note: This code only works when `cols=['MedInc']`, and the model has been trained when that was the case. It will need to be adapted if other features have been used.

```
plt.figure(figsize=(15, 6))
# Plotting MedInc against the Target
plt.scatter(X_train[cols], y_train, s=0.1)

# Selecting x values to input into the model
# Started at the min value of MedInc that we have,
# the upper limit was found by experimenitng so as not to exceed the y value
# in this case the linear model is very poor at predicting anything over a st
andardised x value of over 3
x = np.linspace(X_train[cols].min(), 3.6, 100)

# Using the gradient and intercept from the model to plot
# The intercept is zero but it is added for completeness
# y = m              *x+ c
```

```
y = (regressor.coef_)*x+(regressor.intercept_)

# # Plotting the linear model
plt.plot(x, y, '-r')

# # Adding labels to each axis
plt.xlabel('MedInc Z_scores')
plt.ylabel('Target Z_scores')

# calling the plot
plt.show()
```

Run the below code to plot your model against the training data. How well do you think it fits?

```
plt.figure(figsize=(15, 6))
# Plotting MedInc against the Target
plt.scatter(X_train[cols], y_train, s=0.1)

# Selecting x values to input into the model
# Started at the min value of MedInc that we have,
# the upper limit was found by experimenitng so as not to exceed the y value
# in this case the linear model is very poor at predicting anything over a st
andardised x value of over 3
x = np.linspace(X_train[cols].min(), 3.6, 100)

# Using the gradient and intercept from the model to plot
# The intercept is zero but it is added for completeness
# y = m               *x+ c
y = (regressor.coef_)*x+(regressor.intercept_)

# # Plotting the linear model
plt.plot(x, y, '-r')

# # Adding labels to each axis
plt.xlabel('MedInc Z_scores')
plt.ylabel('Target Z_scores')

# calling the plot
plt.show()
```

Exercise 8 – Evaluating your model

Print the following error metrics for your model. To make them more interpretable, you may wish to invert the scaling of the target variable using `.inverse_transform`:

- `mean_absolute_error`

- `mean_squared_error`
- root mean squared error
- `r2_score`

How well does the model perform? Would you use it?

Exercise 9 – Next steps

What would you do to improve the performance of the model you built?

Extension task

You can try rebuilding the model using the dataset which has had outliers removed to see the difference in model performance, or you could decide which points are outliers yourself and model that data.

Write a brief summary of what impact the removal of outliers has had on the model. Would the model be better or worse at predicting the median price of certain types of houses?

Has its ability to predict one kind of house increased? Has another decreased?