Tutorial 5 Completion Document

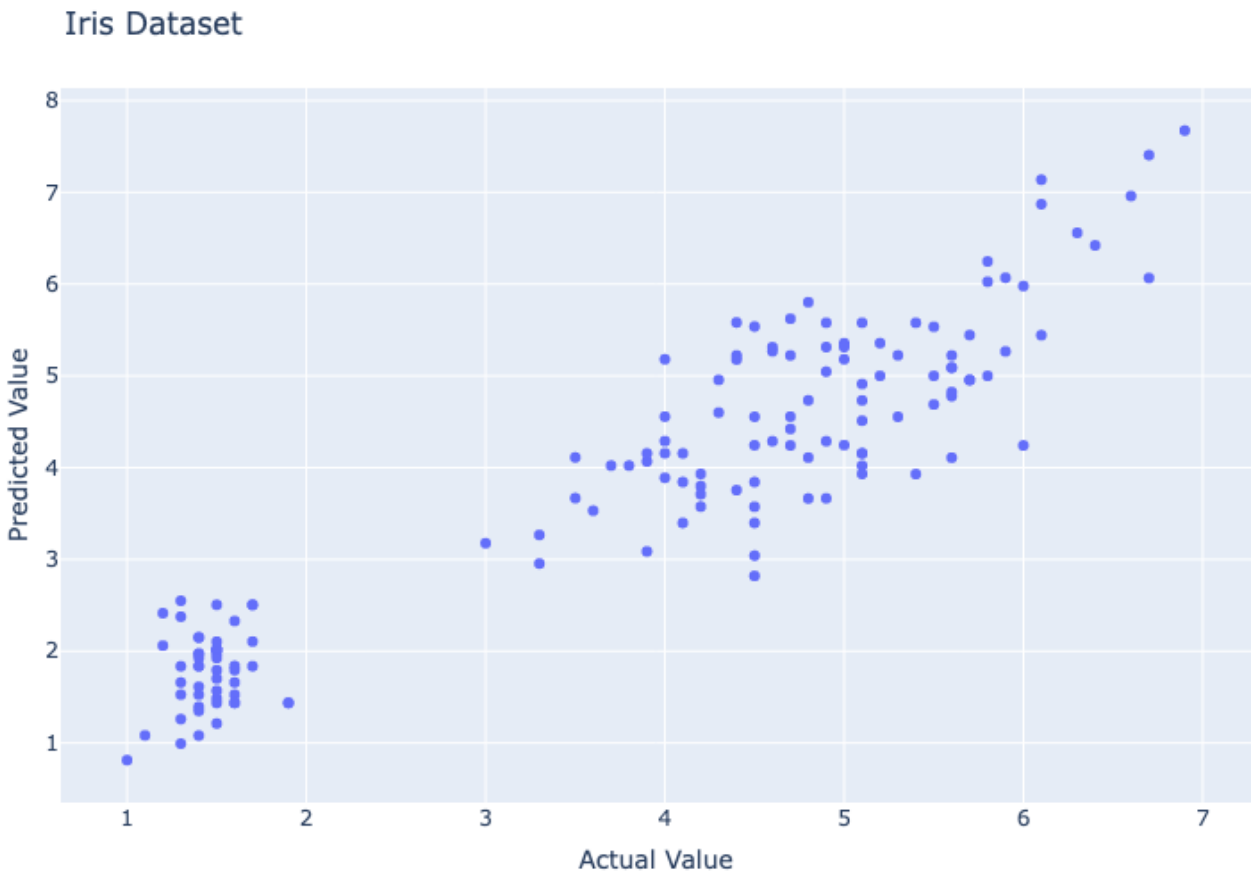
Activity I

In the Tutorial Completion Document, describe your findings.

Fit/train the model with sepal_width and sepal_length and then predict the petal_length. Plot the actual and predicted values using Plotly.

```
import pandas as pd
import plotly.express as px
from sklearn.linear_model import LinearRegression

# Reading the CSV file
df = pd.read_csv(fpath + 'input/iris.csv')

X = df[['sepal_width', 'sepal_length']]
y = df['petal_length']

# Fit/Train the model on sepal width and length, predict the petal length
model = LinearRegression()
model.fit(X, y)

# Predict the petal length for all X values
y_pred = model.predict(X)

# Plot the graph with actual and predicted values
fig = px.scatter(x=y,
                 y=y_pred,
                 labels={'x': 'Actual Value', 'y': 'Predicted Value'})

fig.update_layout(title='Iris Dataset',
                  autosize=False,
                  width=700,
                  height=500,
                  margin=dict(
                      l=0,
```

```
                        r=0,
                        b=0,
                    ), )

fig.show()
```



Description:

I have plotted out the predicted and actual values using the Linear Regression model provided by sckitlearn. Each data point on the graph represents two values: actual and predicted values. Just by eyeballing this graph, I can see that almost every data point lines up vertical with another data point on the graph. The trend of the graph seems to trend upwards. There's a lack of data points between 2-3 inches on the graph, perhaps signifying that there seems to be no data points between those values.

Task I

In the Tutorial Completion Document, describe your findings.

Fit/train the model with the number of `bedrooms` and `bathrooms`. And then, predict the **house price** if the number of **bedrooms is three** and the number of **bathrooms is two**. Plot the actual and predicted values using Plotly.

Note: You need to print the house price (single value) when bedrooms = 3 and bathrooms = 2.

```
import pandas as pd
import plotly.express as px
from sklearn.linear_model import LinearRegression

# Reading the CSV file
df = pd.read_csv(fpath + 'input/HousePrices.csv')

X = df[['bedrooms', 'bathrooms']]
y = df['price']

# Fit/Train the model on bedrooms and bathrooms, predict the house price
model = LinearRegression()
model.fit(X, y)

# Predict the house price for a house with three bedrooms and two bathrooms
y_pred = model.predict([[3, 2]])

# Print the predicted house price
print("Predicted house price: $", round(y_pred[0], 2))

# Plot the graph with actual and predicted values
fig = px.scatter(x=y,
                 y=model.predict(X),
                 labels={'x': 'Actual Value', 'y': 'Predicted Value'})

fig.update_layout(title='Housing Dataset',
                  autosize=False,
                  width=700,
                  height=500,
                  margin=dict(
                      l=0,
                      r=0,
                      b=0,
                  ), )

fig.show()
```

```
# Predicted House Price: $85141.83
```

Description:

It seems that most the the data points on the graph are clustered together. The are no data points from 2-2.99 on the graph. Additionally, there seems to be no discernible outliers that I can observed just by taking an eye's glance. at this graph. It;s hard to see outside of a Juypter environment but the data points on the graph represent both the actual and predicted value. It seems the line of best fit trends upwards and is in close proximity with most of the points on the graph.

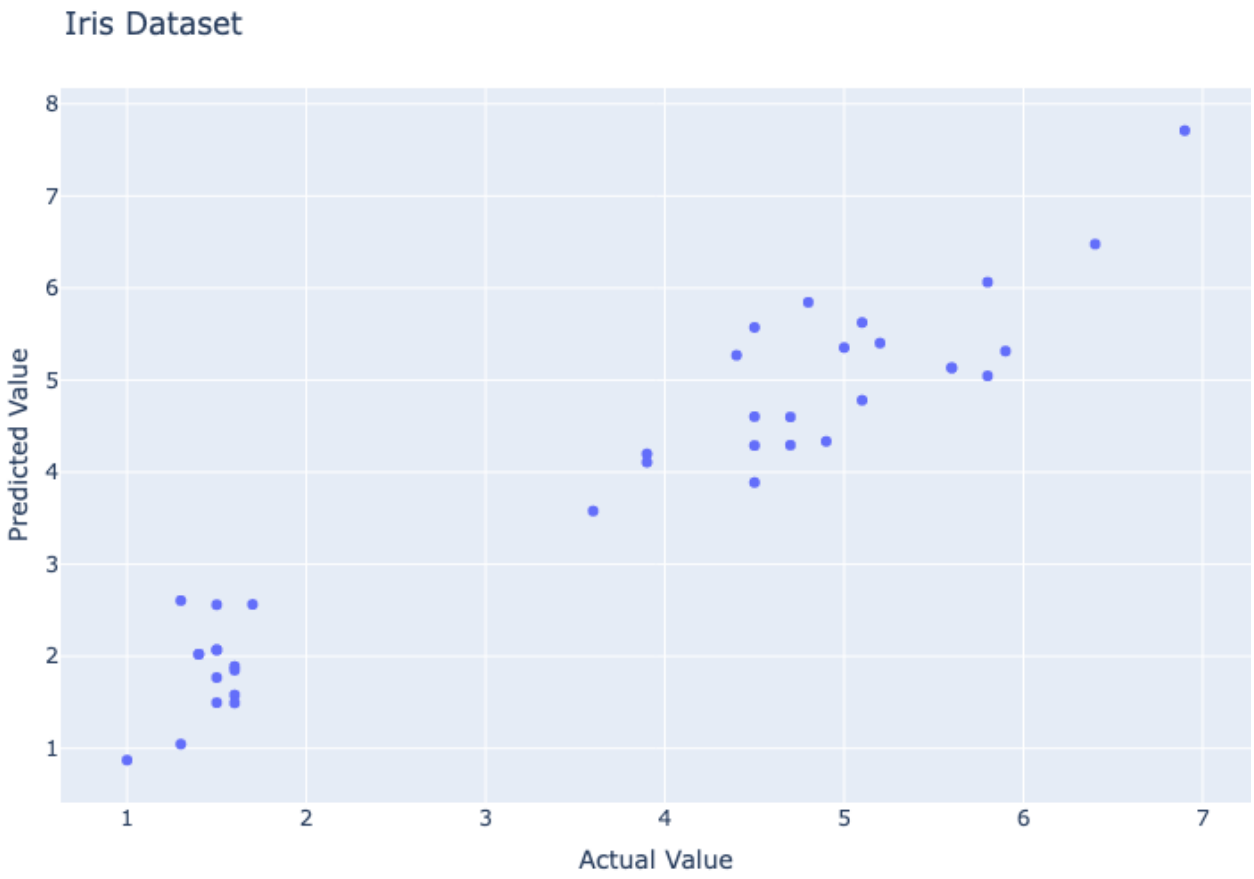
Activity II

In the Tutorial Completion Document, describe your findings.

Report and discuss the three regression evaluation metrics (MAE, MSE, and RMSE) while predicting the petal_length using `sepal_width and sepal_length with 75% train and 25% test data. Plot the actual and predicted values.

```
import pandas as pd
import plotly.express as px
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Reading the CSV file
df = pd.read_csv("data/input/iris.csv")

X = df[['sepal_width', 'sepal_length']]
y = df['petal_length']

# Splitting the dataset into train and test sets with 75% train data and 25% test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Fit/Train the model on sepal width and length, predict the petal length
model = LinearRegression()
model.fit(X_train, y_train)

# Predict the petal length for the test data
y_pred = model.predict(X_test)

# Evaluation Metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

# Print the evaluation metrics
print("MAE: ", round(mae, 2))
print("MSE: ", round(mse, 2))
print("RMSE: ", round(rmse, 2))

# Plot the graph with actual and predicted values
fig = px.scatter(x=y_test,
                 y=y_pred,
                 labels={'x': 'Actual Value', 'y': 'Predicted Value'})

fig.update_layout(title='Iris Dataset',
                  autosize=False,
                  width=700,
```

```
                    height=500,
                    margin=dict(
                        l=0,
                        r=0,
                        b=0,
                    ), )

fig.show()
```



Description:

One thing that sticks out to me right away are the distance between the data points. The distally related data points indicate that there may be other factors at play besides the variables being measured. This type of graph can be useful for identifying positive trends and correlations that may not be immediately apparent. However, scatter plots, similar to other types of graphs, may not fully show all factors at play.

Task II

In the Tutorial Completion Document, describe your findings.

Report and discuss the three regression evaluation metrics (MAE, MSE, and RMSE) while predicting the **house price** using the number of **bedrooms** and the number of **bathrooms** with 80% train and 20% test data. Plot the actual and predicted values.

```
import pandas as pd
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Reading the CSV file
df = pd.read_csv("data/input/HousePrices.csv")

# Selecting the features and target variable
X = df[['bedrooms', 'bathrooms']]
y = df['price']

# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit/Train the model on the training data
model = LinearRegression()
model.fit(X_train, y_train)

# Predict the house price for the test data
y_pred = model.predict(X_test)

# Calculate the regression evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

# Print the metrics
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R-squared:", r2)
```

```
# Plot the actual vs predicted values
fig = px.scatter(x=y_test,
                 y=y_pred,
                 labels={'x': 'Actual Value', 'y': 'Predicted Value'})

fig.update_layout(title='Housing Dataset',
                  autosize=False,
                  width=700,
                  height=500,
                  margin=dict(
                      l=0,
                      r=0,
                      b=0,
                  ), )

fig.show()


"""
MAE: 16965.539529572015
MSE: 517047432.50169194
RMSE: 22738.677017401253
R-squared: 0.22606424840803596

"""
```

Description:

This is a code example in Python that trains a linear regression model on a dataset of house prices and evaluates its performance on a test set using mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) metrics. Additionally, it creates a scatter plot to visualize the actual versus predicted values. The `pandas` library is used to read the data from a CSV file and create a DataFrame. `sklearn` is used to split the data into training and testing sets, train a linear regression model, and evaluate its performance. Finally, `plotly.express` is used to create a scatter plot. The graph generated by this code is a scatter plot. The line of best fit seems to trend upward. However, the data seems to be aggregate towards the lower left corner of the graph. In other words, most of the data points lie between `40K` and `100K` on the actual value of the house price. The fact that are data is not evenly distributed may bias the models' prediction one way or another.