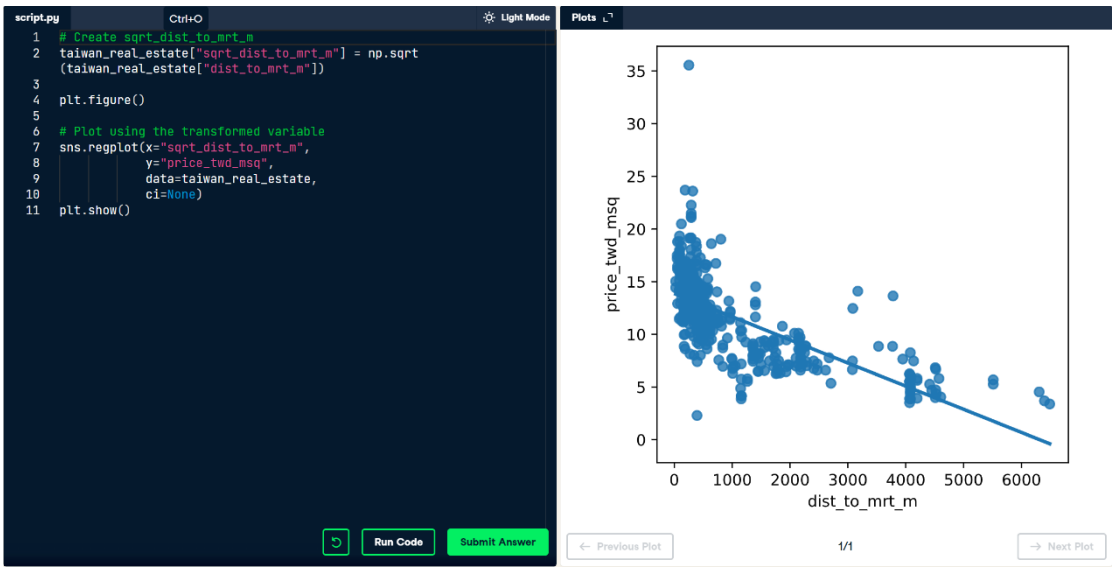
Data Modeling 數據建模
(sns regplot&scatterplot, statemodels.formula.api)

The model before tansform 模型轉換前



將 x 軸用 np.sqrt 平方根後，數值修正，得到正確的分布

The model after tansform 模型轉換後

The Comparison of Original and Predict model

利用上述模型，與預測模型(線上紅點)做比較，觀察結果跟預測數值差異

紅點皆在點上，表示預測值與實際結果相符

當周圍的便利商店數量多，房地坪數價格會相對高

RSE of original model and transformed model

RSE 在模型轉換後，值變小，代表模型準確率提升

script.py

```python
# Calculate mse_orig for mdl_click_vs_impression_orig
mse_orig = mdl_click_vs_impression_orig.mse_resid

# Calculate rse_orig for mdl_click_vs_impression_orig and print it
rse_orig = np.sqrt(mse_orig)
print("RSE of original model: ", rse_orig)

# Calculate mse_trans for mdl_click_vs_impression_trans
mse_trans = mdl_click_vs_impression_trans.mse_resid

# Calculate rse_trans for mdl_click_vs_impression_trans and print it
rse_trans = np.sqrt(mse_trans)
print("RSE of transformed model: ", rse_trans)
```

Light Mode

Run Code

Submit Answer

IPython Shell | Slides | Notes

```
# Calculate and print the specificity
specificity = TN / (TN + FP )
print("specificity: ", specificity)

# Calculate mse_orig for mdl_click_vs_impression_orig
mse_orig = mdl_click_vs_impression_orig.mse_resid

# Calculate rse_orig for mdl_click_vs_impression_orig and print it
rse_orig = np.sqrt(mse_orig)
print("RSE of original model: ", rse_orig)

# Calculate mse_trans for mdl_click_vs_impression_trans
mse_trans = mdl_click_vs_impression_trans.mse_resid

# Calculate rse_trans for mdl_click_vs_impression_trans and print it
rse_trans = np.sqrt(mse_trans)
print("RSE of transformed model: ", rse_trans)

RSE of original model:  19.905838862478138
RSE of transformed model:  0.19690640896875722
```

The Comparison of Original and Logistic Predict Model

預測模型(藍色)在 logistic 後變成曲線，尾端拉升

代表客戶距離第一次購買產品(time_since_first_purchase)，時間隔得越久

客戶越容易流失 (has_churned = 0~1 未流失~流失)

將預測模型轉為 scatterplot

將模型改為另一種簡單預測(紅點)

Most Likely Outcome

(has_churned, 即<0.5 客戶不會流失，>0.5 客戶更有可能會流失)

可得出距離首次購買時間間隔越久，流失機率越高

script.py

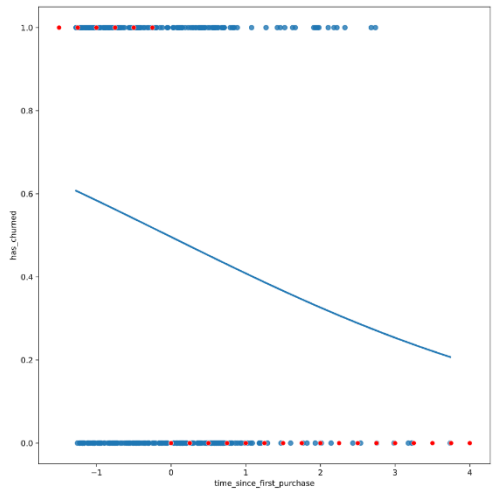
```python
# Update prediction data by adding most_likely_outcome
prediction_data["most_likely_outcome"] = np.round(prediction_data
["has_churned"])

fig = plt.figure()

# Create a scatter plot with logistic trend line (from previous
exercise)
sns.regplot(x="time_since_first_purchase",
            y="has_churned",
            data=churn,
            ci=None,
            logistic=True)

# Overlay with prediction_data, colored red
sns.scatterplot(x='time_since_first_purchase',
                y='most_likely_outcome',
                data=prediction_data,
                color='red')

plt.show()
```

Light Mode

Run Code

Submit Answer

Plots



Previous Plot

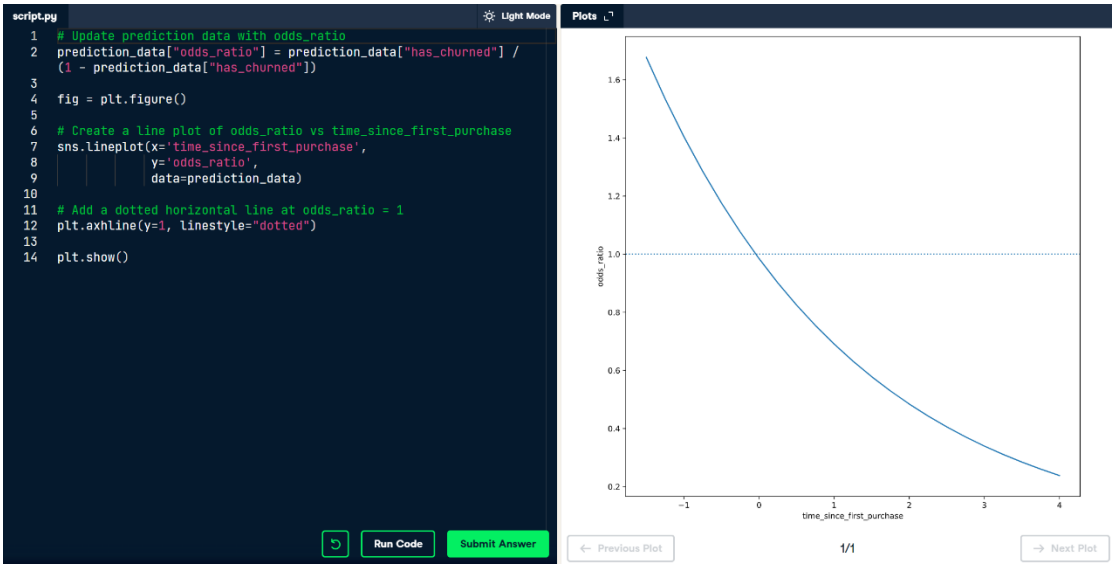
1/1

Next Plot

The Comparison of Odds Ration and the Time First Purchase

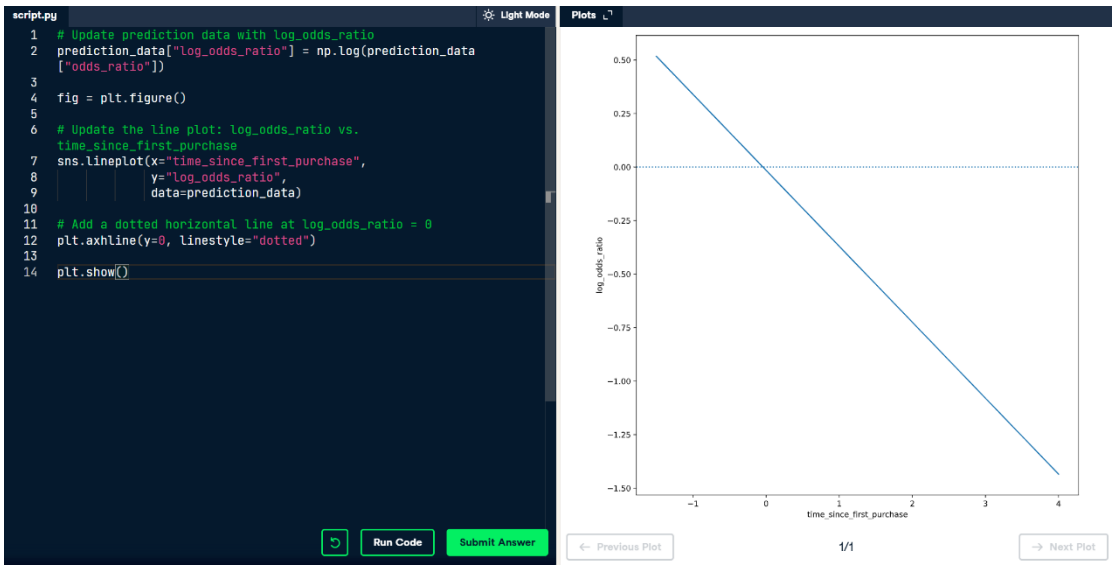
利用 Odds Ratio 賠率比與首次購買時間做比較

距離首購時間越久，賠率比越接近 1，流失率越高



將賠率比 log，使模型穩定

沒有 log 的賠率比(即曲線)，模型波動較大，不易穩定



Linear Regression 線性回歸分類比較

script.py

```python
# Update the model formula to remove the intercept
mdl_price_vs_age0 = ols("price_twd_msq ~ house_age_years + 0", data=taiwan_real_estate).fit()

# Print the parameters of the fitted model
print(mdl_price_vs_age0.params)
```

Run Code Submit Answer

IPython Shell Slides Notes

```
house_age_years[T.30 to 45]    -1.244
dtype: float64


<script.py> output:
    Intercept                     12.637
    house_age_years[T.15 to 30]   -2.761
    house_age_years[T.30 to 45]   -1.244
    dtype: float64

# Update the model formula to remove the intercept
mdl_price_vs_age0 = ols("price_twd_msq ~ house_age_years + 0", data=taiwan_real_estate).fit()

# Print the parameters of the fitted model
print(mdl_price_vs_age0.params)

house_age_years[0 to 15]     12.637
house_age_years[15 to 30]     9.877
house_age_years[30 to 45]    11.393
dtype: float64

In [1]:
```

Confusion Matrix 模糊矩陣

script.py

```python
# Get the actual responses
actual_response = churn['has_churned']

# Get the predicted responses
predicted_response = np.round(mdl_churn_vs_relationship.predict())

# Create outcomes as a DataFrame of both Series
outcomes = pd.DataFrame({"actual_response": actual_response,
                         "predicted_response":predicted_response})

# Print the outcomes
print(outcomes.value_counts(sort = False))
```

Light Mode

Run Code

Submit Answer

script.py solution.py

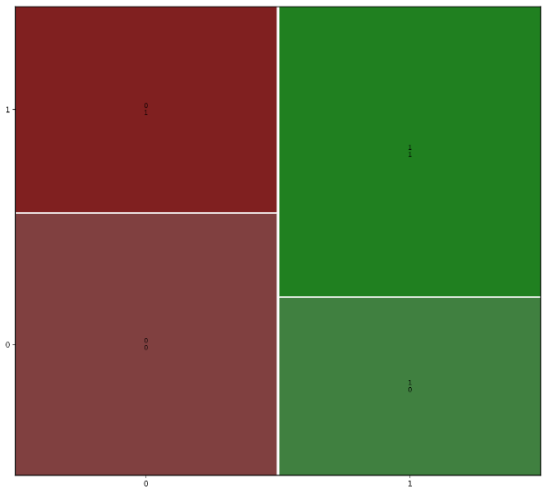
Light Mode

```python
# Import mosaic from statsmodels.graphics.mosaicplot
from statsmodels.graphics.mosaicplot import mosaic

# Calculate the confusion matrix conf_matrix
conf_matrix = mdl_churn_vs_relationship.pred_table()

# Print it
print(conf_matrix)

# Draw a mosaic plot of conf_matrix
mosaic(conf_matrix)
plt.show()
```

Run Code

Submit Answer

Plots



Previous Plot

1/1

Next Plot