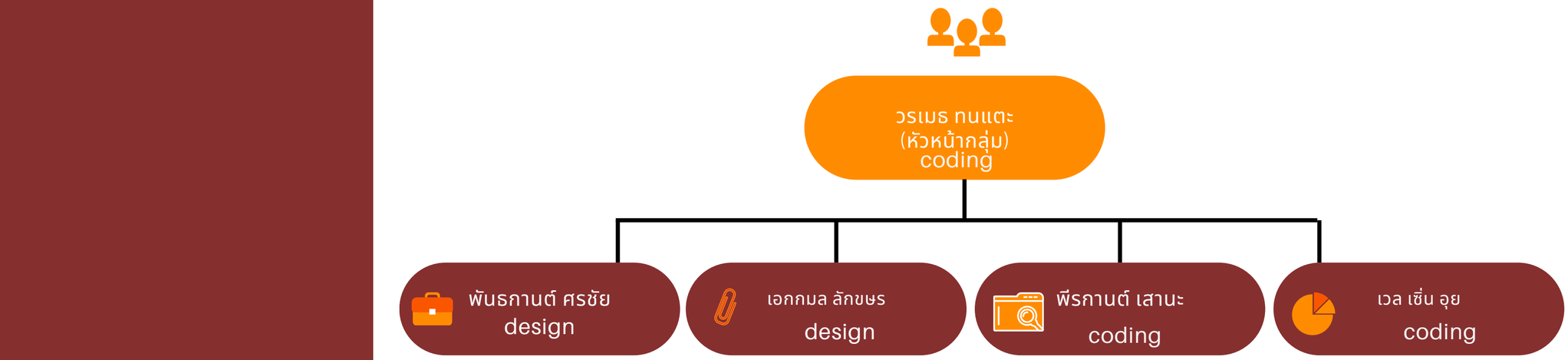
# BAMBOO

Brazilian E-Commerce

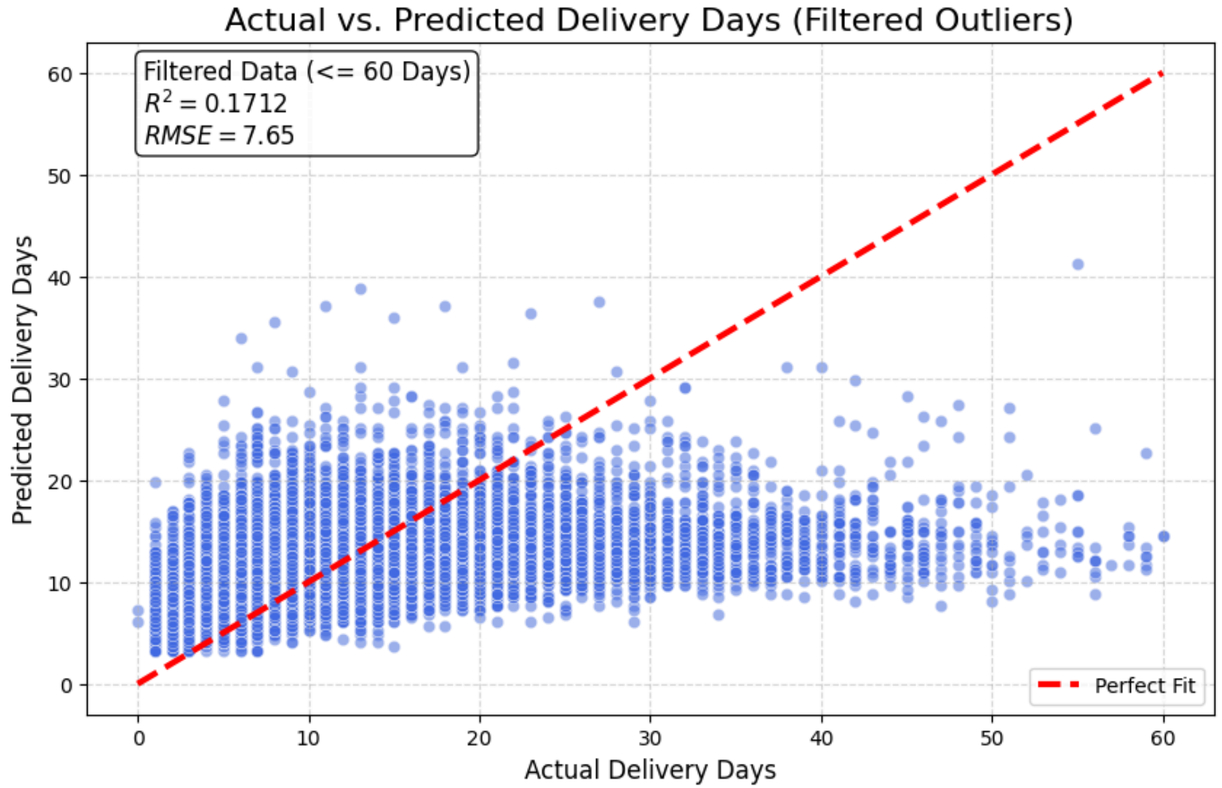
# BAMBOO

Group members



# GRAPH

```
Original Data Size: 99441
Filtered Data Size: 96188
```




แกน x = วันที่ส่งจริง (Actual Delivery Days)

แกน Y = วันที่โมเดลคาดการณ์ (Predicted Delivery Days)

## $R^2$ = 0.1712

โมเดลนี้ มีค่าที่ได้คือ 0.17 ซึ่งเป็นค่าที่ถือมาก
น้อยมากแม่นยำแค่ 17%

## RMSE = 7.65

โมเดล Linear Regression อธิบายความสัมพันธ์ระหว่าง
estimated_days กับ delivery_days ได้น้อยมาก

ค่าเฉลี่ยความคลาดเคลื่อนของการทำนาย ≈ ผิดประมาณ 7.6 วัน

# CODE

```python
# 1. กรองข้อมูล (Remove Outliers > 60 days)
# ใช้ .copy() เพื่อป้องกัน SettingWithCopyWarning
df_filtered = BZ_order_dataset.dropna(subset=['delivery_days', 'estimated_days']).copy()
df_filtered = df_filtered[df_filtered['delivery_days'] <= 60]

print(f"Original Data Size: {len(BZ_order_dataset)}")
print(f"Filtered Data Size: {len(df_filtered)}")
```

**ลบข้อมูลที่มีค่าว่าง (missing) ในคอลัมน์**
delivery_days
estimated_days

**ลบข้อมูลที่เป็น Outlier**
เอาเฉพาะแถวที่ delivery_days ไม่เกิน 60 วัน
นับจำนวนแถวก่อนและหลังการกรอง เพื่อดูว่าเหลือข้อมูลกี่แถว

```python
# 2. Prepare X and y
X = df_filtered[['estimated_days']]
y = df_filtered['delivery_days']

# 3. Train-Test Split (80:20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Train Model (Linear Regression)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# 5. Calculate Metrics
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
```

เลือกข้อมูล X = estimated_days และ y = delivery_days เพื่อใช้ฝึกโมเดล
แบ่งข้อมูลเป็นชุดฝึก 80% และชุดทดสอบ 20% แล้วสร้างโมเดล

**Linear Regression**
ทำนายผลและคำนวณความแม่นยำด้วยค่า RMSE และ $R^2$

```python
# 6. Visualization
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.5, color='royalblue', edgecolor='w')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=3, label='Perfect Fit')

# ใส่ค่า Metrics ลงในกราฟ
plt.text(0.05, 0.85, f'Filtered Data (<= 60 Days)\n$R^2 = {r2:.4f}$\n$RMSE = {rmse:.2f}$',
         transform=plt.gca().transAxes, fontsize=12,
         bbox=dict(boxstyle='round', facecolor='white', alpha=0.9))

plt.title('Actual vs. Predicted Delivery Days (Filtered Outliers)', fontsize=16)
plt.xlabel('Actual Delivery Days', fontsize=12)
plt.ylabel('Predicted Delivery Days', fontsize=12)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

**สร้างกราฟ** scatter แสดงความสัมพันธ์ระหว่างค่าส่งจริง (y_test) และค่าที่โมเดลทำนาย (y_pred)
วาดเส้นสีแดงเพื่อเป็นเส้นอ้างอิงว่าถ้าทำนายถูก 100% ต้องอยู่บนเส้นนี้
ใส่กล่องข้อความแสดงค่า $R^2$ และ RMSE บนกราฟ
ตั้งชื่อกราฟ ชื่อแกน X-Y และแสดง legend
เปิดเส้นกริดและสั่งแสดงกราฟด้วย plt.show()

# Classification

## features

```
# 4. สร้าง Features (ตัวแปรต้น)
# สิ่งที่มีผลต่อความพอใจ: ส่งช้าไหม?, ราคาแพงไหม?, ค่าส่งแพงไหม?
df['order_purchase_timestamp'] = pd.to_datetime(df['order_purchase_timestamp'])
df['order_delivered_customer_date'] = pd.to_datetime(df['order_delivered_customer_date'])
df['order_estimated_delivery_date'] = pd.to_datetime(df['order_estimated_delivery_date'])

# Feature : ส่งช้ากว่าที่สัญญาไว้กี่วัน? (ค่าติดลบแปลว่าส่งก่อนกำหนด = ดี)
df['delay_days'] = (df['order_delivered_customer_date'] - df['order_estimated_delivery_date']).dt.days

feature_cols = [
    'price',
    'freight_value',
    'delay_days'
]
```

# LIBRARY


**lazypredict**

Lazy Predict help build a lot of basic models without much code and helps understand which models works better without any parameter tuning

PyPI

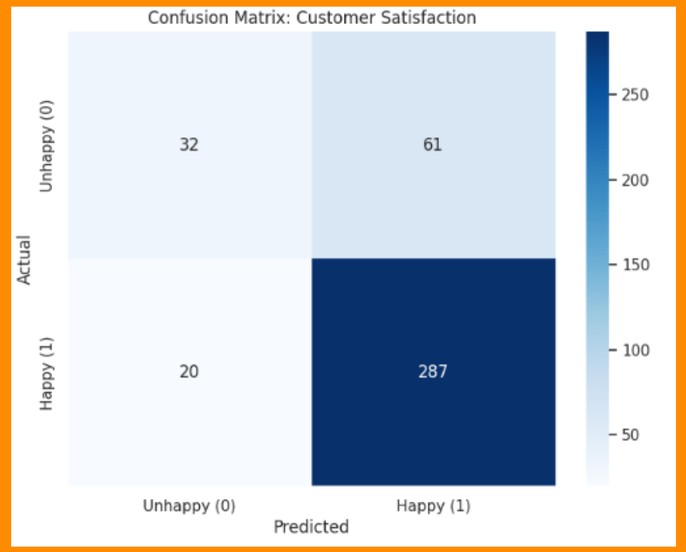
ใช้ library ตัวนี้เพื่อเปรียบเทียบ

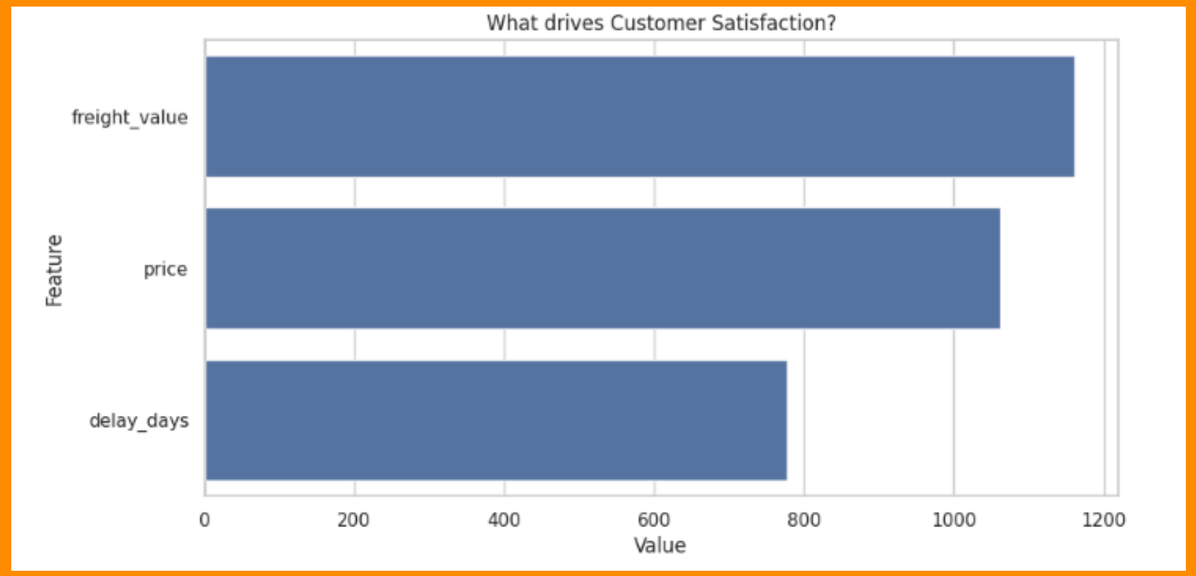
แต่ละตัวโมเดลตัวไหนดีสุด

| Model | Accuracy | Balanced Accuracy | ROC AUC | F1 Score | \ |
|---|---|---|---|---|---|
| BaggingClassifier | 0.81 | 0.70 | 0.70 | 0.80 | |
| RandomForestClassifier | 0.83 | 0.69 | 0.69 | 0.82 | |
| ExtraTreesClassifier | 0.83 | 0.69 | 0.69 | 0.81 | |
| DecisionTreeClassifier | 0.76 | 0.67 | 0.67 | 0.76 | |
| LGBMClassifier | 0.82 | 0.67 | 0.67 | 0.80 | |
| XGBClassifier | 0.80 | 0.66 | 0.66 | 0.78 | |
| ExtraTreeClassifier | 0.76 | 0.65 | 0.65 | 0.76 | |
| QuadraticDiscriminantAnalysis | 0.80 | 0.61 | 0.61 | 0.76 | |
| LabelSpreading | 0.80 | 0.61 | 0.61 | 0.76 | |
| GaussianNB | 0.79 | 0.60 | 0.60 | 0.76 | |
| LabelPropagation | 0.79 | 0.60 | 0.60 | 0.76 | |
| KNeighborsClassifier | 0.78 | 0.60 | 0.60 | 0.75 | |
| AdaBoostClassifier | 0.81 | 0.60 | 0.60 | 0.76 | |
| SVC | 0.80 | 0.57 | 0.57 | 0.74 | |
| PassiveAggressiveClassifier | 0.71 | 0.57 | 0.57 | 0.70 | |
| Perceptron | 0.64 | 0.54 | 0.54 | 0.65 | |
| SGDClassifier | 0.77 | 0.54 | 0.54 | 0.70 | |
| NearestCentroid | 0.59 | 0.53 | 0.53 | 0.62 | |
| LogisticRegression | 0.78 | 0.52 | 0.52 | 0.68 | |
| LinearDiscriminantAnalysis | 0.77 | 0.51 | 0.51 | 0.68 | |
| CalibratedClassifierCV | 0.77 | 0.51 | 0.51 | 0.68 | |
| LinearSVC | 0.77 | 0.51 | 0.51 | 0.67 | |
| RidgeClassifierCV | 0.77 | 0.51 | 0.51 | 0.67 | |
| RidgeClassifier | 0.77 | 0.51 | 0.51 | 0.67 | |
| BernoulliNB | 0.77 | 0.50 | 0.50 | 0.67 | |
| DummyClassifier | 0.77 | 0.50 | 0.50 | 0.67 | |

| Model | Time Taken |
|---|---|
| BaggingClassifier | 0.11 |
| RandomForestClassifier | 0.37 |
| ExtraTreesClassifier | 0.24 |
| DecisionTreeClassifier | 0.03 |
| LGBMClassifier | 0.08 |
| XGBClassifier | 0.08 |
| ExtraTreeClassifier | 0.02 |
| QuadraticDiscriminantAnalysis | 0.02 |
| LabelSpreading | 0.19 |
| GaussianNB | 0.02 |
| LabelPropagation | 0.12 |
| KNeighborsClassifier | 0.02 |
| AdaBoostClassifier | 0.18 |
| SVC | 0.11 |
| PassiveAggressiveClassifier | 0.02 |
| Perceptron | 0.04 |
| SGDClassifier | 0.02 |
| NearestCentroid | 0.02 |
| LogisticRegression | 0.02 |
| LinearDiscriminantAnalysis | 0.02 |
| CalibratedClassifierCV | 0.06 |
| LinearSVC | 0.03 |
| RidgeClassifierCV | 0.02 |
| RidgeClassifier | 0.02 |
| BernoulliNB | 0.03 |
| DummyClassifier | 0.02 |

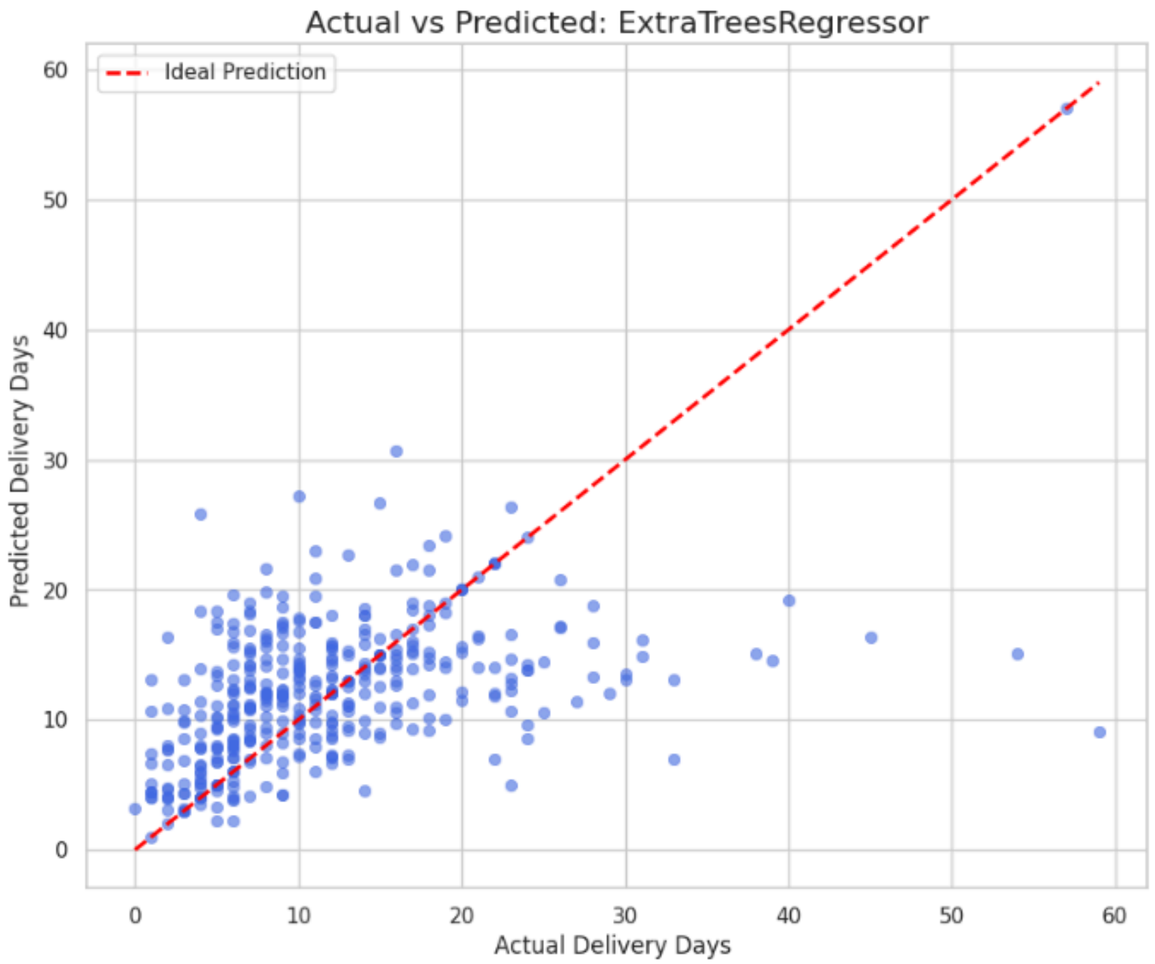
# LGBMClassifier




## อะไรทำให้ลูกค้า Happy/Unhappy?

จากผลลัพธ์ที่แสดงผลมาเราจะเห็นได้ว่า freight value (มูลค่าการขนส่งสินค้า) มีผลต่อความพึงพอใจลูกค้ามากที่สุดส่วนที่เหลือรองลงมา ตามกราฟในรูป




จุดที่อยู่เหนือเส้นสีแดง (Y > X): โมเดล ทำนายสูงเกินไป (Overestimate) เช่น วันจริง 5 วัน แต่ทำนายเป็น 10 วัน

จุดที่อยู่ใต้เส้นสีแดง (Y < X): โมเดล ทำนายต่ำเกินไป (Underestimate) เช่น วันจริง 20 วัน แต่ทำนายเป็น 10 วัน

# CODE

```
# 1. สร้างโมเดลใหม่ (ใช้ตัวที่เก่งที่สุดจากผลลัพธ์ของคุณ)
# สมมติว่าจาก error ของคุณ ตัวที่เก่งสุดคือ 'ExtraTreesRegressor'
reg_final = ExtraTreesRegressor(random_state=42)

# หรือถ้าตัวอื่นชนะ ให้เปลี่ยนชื่อฟังก์ชันตรงนี้ เช่น:
# reg_final = LGBMRegressor()
# reg_final = XGBRegressor()
```

```
# 2. สอนโมเดล (Train)
print(f"กำลังเทรนโมเดล: {type(reg_final).__name__} ...")
reg_final.fit(X_train, y_train)

# 3. ให้โมเดลลองทำนาย (Predict)
y_pred_final = reg_final.predict(X_test)
```

2. Train

- **การสุ่มคุณลักษณะ (Feature Randomness):** เช่นเดียวกับ Random Forest, Extra Trees จะสุ่มคุณลักษณะย่อยๆ มาพิจารณาในการแบ่งแต่ละโหนด
- **การสุ่มจุดแบ่ง (Split Randomness):** ในขณะที่ Random Forest จะหาจุดแบ่งที่ดีที่สุด (Optimal Split) จากคุณลักษณะที่เลือก Extra Trees จะเลือกจุดแบ่งแบบสุ่ม (Random Split) มาใช้เลย
  - การทำเช่นนี้ทำให้เกิดความผันผวน (Variance) ที่สูงขึ้นเล็กน้อย แต่ช่วยลด Bias และลด Overfitting

3 . Predict

- **.fit() :** เป็นเมธอดมาตรฐานที่ใช้ในการฝึกฝนโมเดลในไลบรารี scikit-learn
- **X_train :** คือ ชุดข้อมูลคุณลักษณะ (Features) สำหรับการฝึกฝน ประกอบด้วยตัวแปรอิสระที่ใช้ในการทำนาย (เช่น วันที่สั่งซื้อ, หมวดหมู่สินค้า, ที่ตั้ง ฯลฯ)
- **y_train :** คือ ชุดข้อมูลเป้าหมาย (Target) สำหรับการฝึกฝน ประกอบด้วยค่าจริงที่เราต้องการให้โมเดลเรียนรู้การทำนาย (ในกรณีนี้คือ "Actual Delivery Days")

- **.predict( ) :** เป็นเมธอดที่สั่งให้โมเดลใช้สิ่งที่เรียนรู้มาในการทำนายค่าเป้าหมาย
- **X_test :** คือ ชุดข้อมูลคุณลักษณะ (Features) สำหรับการทดสอบ เป็นข้อมูลที่คุณลักษณะเหมือนกับ X_train แต่ ค่าเป้าหมายจริง (y_test) ไม่ได้ถูกเปิดเผยให้โมเดลเห็น ในระหว่างการฝึกฝน
- **y_pred_final :** เป็นตัวแปรที่เก็บ ค่าที่ทำนายได้ จากโมเดลสำหรับข้อมูลใน X_test ค่าเหล่านี้จะถูกนำไปเปรียบเทียบกับค่าจริง (y_test) ในขั้นตอนถัดไป (เช่น การสร้างกราฟ Actual vs Predicted หรือการคำนวณ Error Metrics)

# CODE

```
# 4. สร้างกราฟ Scatter Plot
plt.figure(figsize=(10, 8))
sns.set_theme(style="whitegrid")

# พล็อตจุด
sns.scatterplot(x=y_test, y=y_pred_final, alpha=0.6, edgecolor=None, color='royalblue')
```

- Canvas Setup เป็นการสร้าง "ผืนผ้าใบ" หรือ "Figure" สำหรับวาดกราฟจากไลบรารี Matplotlib ,กำหนดขนาดของกราฟ

- Plotting the Points
  - sns.scatterplot(x=y_test, y=y_pred_final, alpha=0.6, edgecolor=None, color='royalblue')
  - นี่คือคำสั่งหลักในการสร้างกราฟ Scatter Plot
  - sns.scatterplot(): ฟังก์ชันของ Seaborn สำหรับสร้างกราฟการกระจายจุด
  - x=y_test: กำหนดให้ค่าจริง (Actual Delivery Days) อยู่บนแกน X
  - y=y_pred_final: กำหนดให้ค่าที่ทำนาย (Predicted Delivery Days) อยู่บนแกน Y
  - alpha=0.6: กำหนด ความโปร่งใส (Opacity) ของจุด จุดจะมีสีจางลง 40% (ทึบ 60%) ซึ่งมีประโยชน์เมื่อมีจุดข้อมูลจำนวนมากซ้อนทับกัน จะช่วยให้มองเห็นความหนาแน่นของจุดได้
  - edgecolor=None: ลบเส้นขอบรอบจุดข้อมูลออก
  - color='royalblue': กำหนดสีของจุดข้อมูลเป็นสีน้ำเงินเข้ม

# CODE

```
# สร้างเส้นทแยงมุมเทียบความแม่น
min_val = min(y_test.min(), y_pred_final.min())
max_val = max(y_test.max(), y_pred_final.max())
plt.plot([min_val, max_val], [min_val, max_val], color='red', linestyle='--', linewidth=2, label='Ideal Prediction')

plt.title(f"Actual vs Predicted: {type(reg_final).__name__}", fontsize=16)
plt.xlabel("Actual Delivery Days ", fontsize=12)
plt.ylabel("Predicted Delivery Days ", fontsize=12)
plt.legend()
plt.show()
```

- min_val = min(y_test.min(), y_pred_final.min())
  - คำนวณหา ค่าต่ำสุด ของทั้งค่าจริง (y_test) และค่าที่ทำนาย (y_pred_final) เพื่อใช้เป็นจุดเริ่มต้นของเส้นทแยงมุม
- max_val = max(y_test.max(), y_pred_final.max())
  - คำนวณหา ค่าสูงสุด ของทั้งค่าจริงและค่าที่ทำนาย เพื่อใช้เป็นจุดสิ้นสุดของเส้นทแยงมุม
- plt.plot([min_val, max_val], [min_val, max_val], color='red', linestyle='--', linewidth=2, label='Ideal Prediction')
  - ใช้ฟังก์ชัน plt.plot() ของ Matplotlib เพื่อวาดเส้นตรง
  - [min_val, max_val], [min_val, max_val]: เป็นการระบุพิกัด (X, Y) โดยให้พิกัด X เริ่มจาก min_val ไป max_val และพิกัด Y ก็เริ่มจาก min_val ไป max_val เช่นกัน ซึ่งรับประกันว่าเส้นที่วาดจะเป็นเส้นตรง color='red': กำหนดให้เส้นเป็นสีแดง
  - linestyle='--': กำหนดให้เส้นเป็นเส้นประ
  - linewidth=2: กำหนดความหนาของเส้น
  - label='Ideal Prediction': กำหนดชื่อสำหรับเส้นนี้ เพื่อใช้แสดงในคำอธิบายกราฟ (Legend)