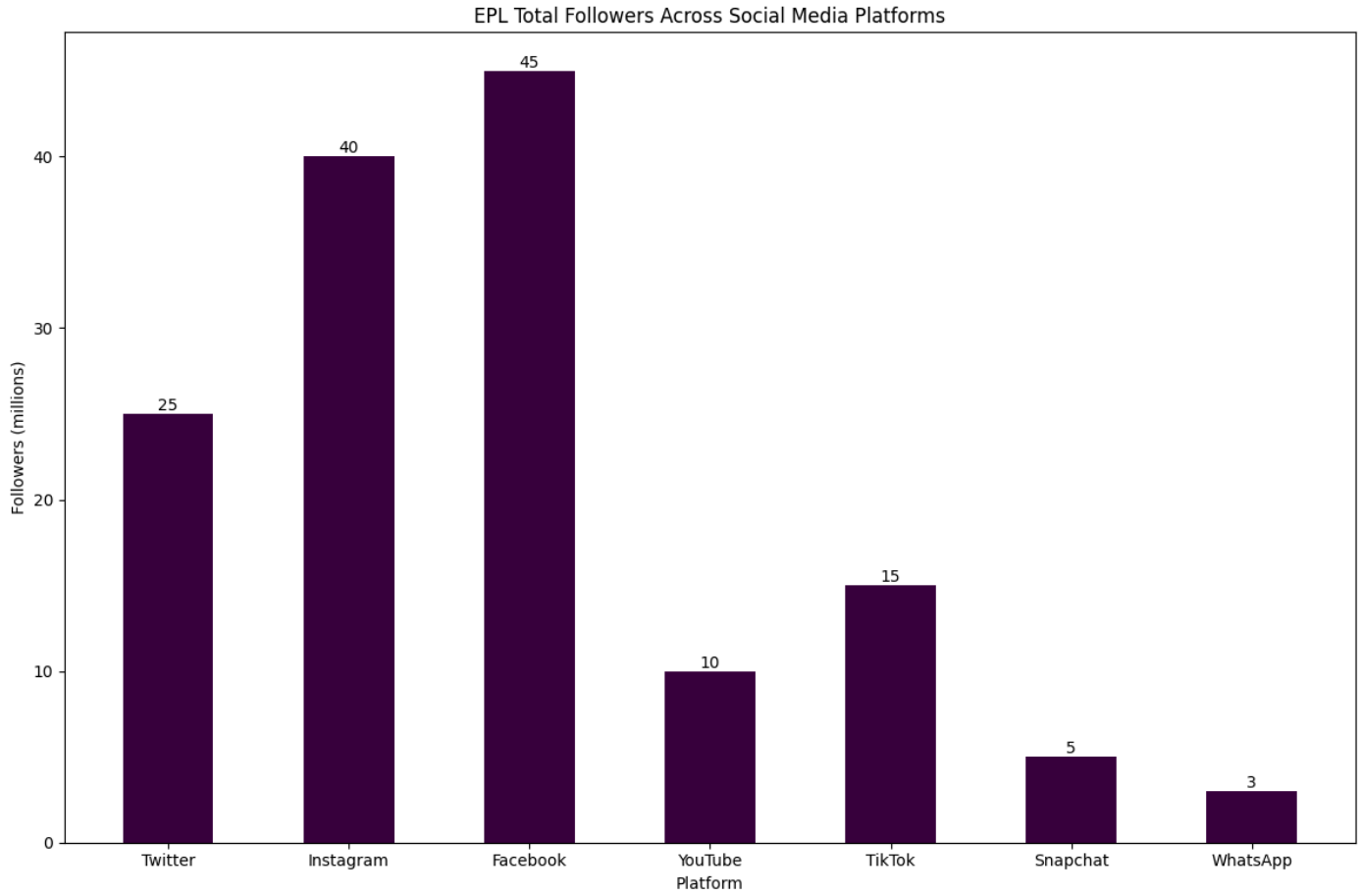
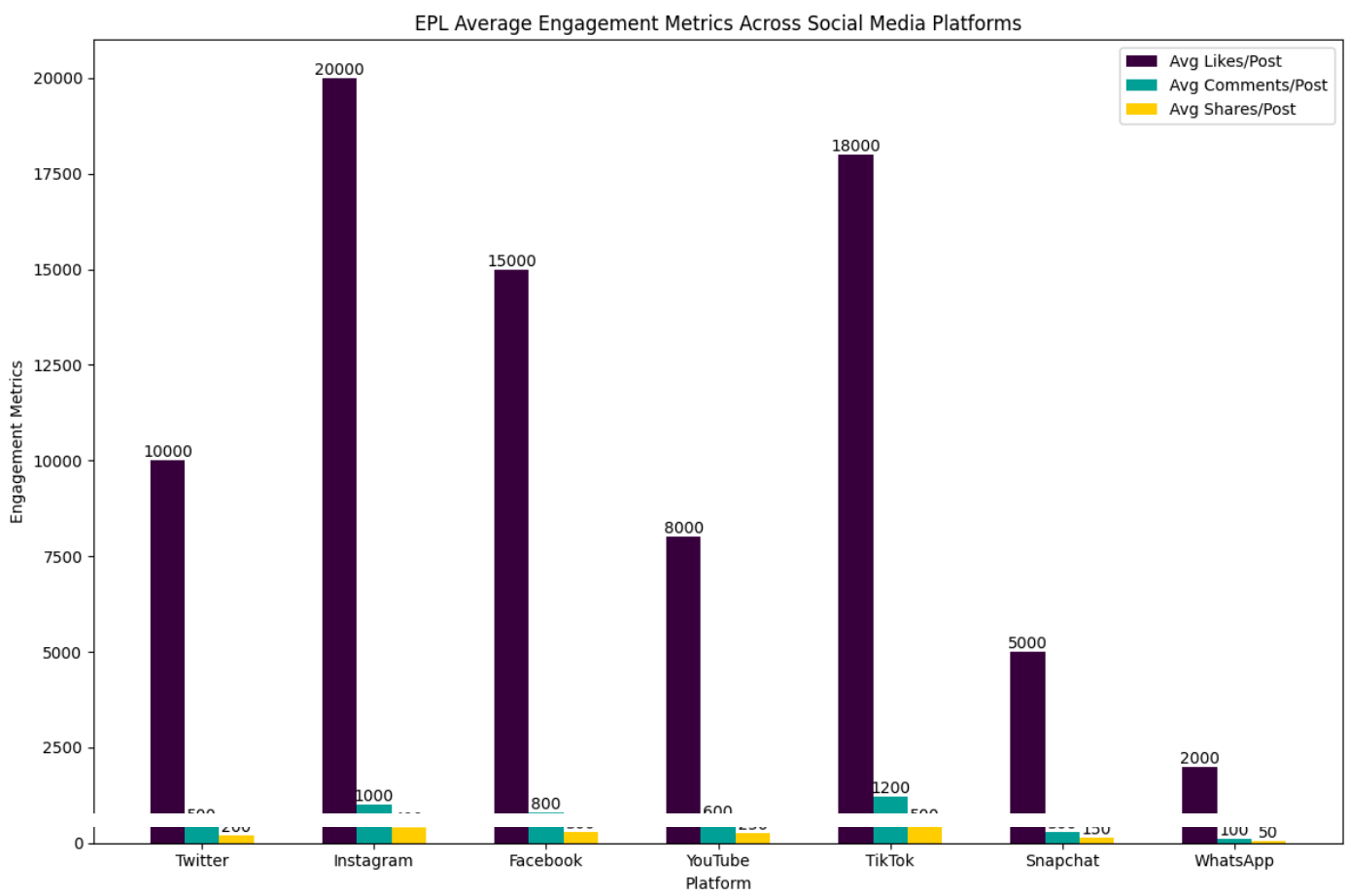
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc


platforms = ['Twitter', 'Instagram', 'Facebook', 'YouTube', 'TikTok', 'Snapchat', 'WhatsApp']
followers = [25, 40, 45, 10, 15, 5, 3]

colors = '#37003c'

fig, ax = plt.subplots(figsize=(12, 8))

bar_width = 0.5
index = range(len(platforms))

bars = plt.bar(index, followers, bar_width, color=colors)

plt.xlabel('Platform')
plt.ylabel('Followers (millions)')
plt.title('EPL Total Followers Across Social Media Platforms')
plt.xticks(index, platforms)

def add_labels(bars):
    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2., height,
                 f'{height:.0f}', ha='center', va='bottom')

add_labels(bars)

plt.tight_layout()
plt.show()
```

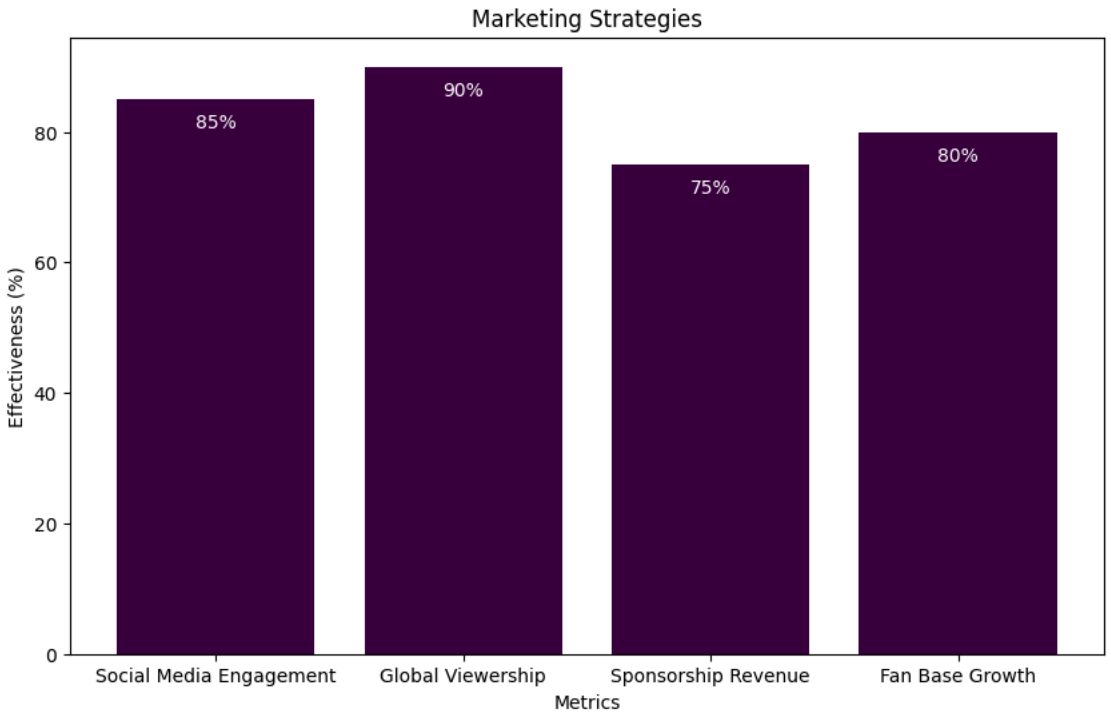



```
platforms = ['Twitter', 'Instagram', 'Facebook', 'YouTube', 'TikTok', 'Snapchat', 'WhatsApp']
avg_likes = [10000, 20000, 15000, 8000, 18000, 5000, 2000]
avg_comments = [500, 1000, 800, 600, 1200, 300, 100]
avg_shares = [200, 400, 300, 250, 500, 150, 50]

colors = {
    'avg_likes': '#37003c',
    'avg_comments': '#00a398',
    'avg_shares': '#ffcd00'
}

fig, ax = plt.subplots(figsize=(12, 8))

bar_width = 0.2
index = range(len(platforms))

bar1 = plt.bar(index, avg_likes, bar_width, label='Avg Likes/Post', color=colors['avg_likes'])
bar2 = plt.bar([i + bar_width for i in index], avg_comments, bar_width, label='Avg Comments/Post', color=colors['avg_comments'])
bar3 = plt.bar([i + bar_width * 2 for i in index], avg_shares, bar_width, label='Avg Shares/Post', color=colors['avg_shares'])

plt.xlabel('Platform')
plt.ylabel('Engagement Metrics')
plt.title('EPL Average Engagement Metrics Across Social Media Platforms')
plt.xticks([i + bar_width for i in index], platforms)
plt.legend()

def add_labels(bars):
    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2., height,
                 f'{height:.0f}', ha='center', va='bottom')

add_labels(bar1)
```

```
add_labels(bar2)
add_labels(bar3)

plt.tight_layout()
plt.show()
```




```
metrics = ['Social Media Engagement', 'Global Viewership', 'Sponsorship Revenue', 'Fan Base Growth']
values = [85, 90, 75, 80]

epl_purple = '#37003c'

plt.figure(figsize=(10, 6))
bars = plt.bar(metrics, values, color=epl_purple)

plt.title('Marketing Strategies')
plt.xlabel('Metrics')
plt.ylabel('Effectiveness (%)')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2.0, yval - 5, f'{yval}%', ha='center', va='bottom', color='white')

plt.show()
```

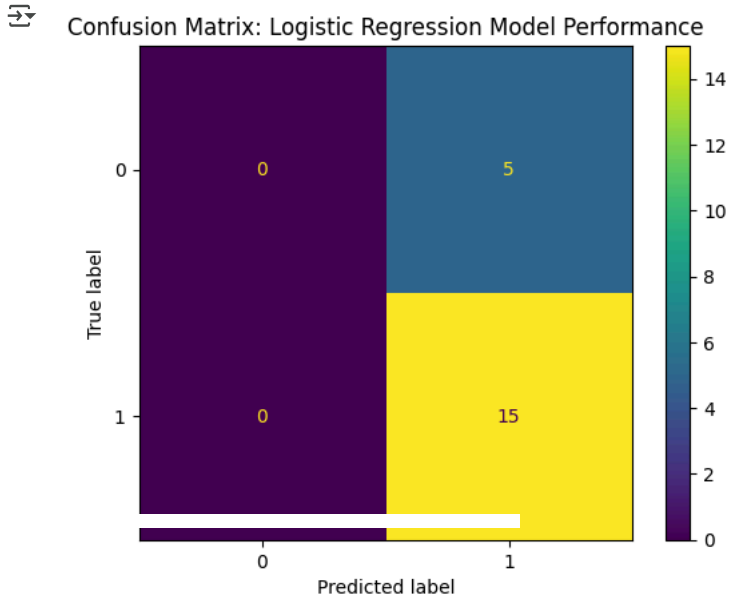



```
data = {
    "Matchday (£m)": [82, 604, 583, 622, 620, 670, 683, 599, 31, 763, 867, 871, 932],
    "Broadcasting (£m)": [1758, 1780, 1927, 2768, 2844, 3049, 2331, 3345, 2954, 3232, 3173, 3352, 3520],
    "Commercial (£m)": [73, 897, 987, 1090, 1168, 1305, 1418, 1563, 1493, 1738, 1960, 1975, 2100]
}

df = pd.DataFrame(data)

df["Total Revenue"] = df.sum(axis=1)
df["High_Revenue_Year"] = (df["Total Revenue"] > 4000).astype(int)

X = df[["Matchday (£m)", "Broadcasting (£m)", "Commercial (£m)"]]
y = df["High_Revenue_Year"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

model = LogisticRegression()
model.fit(X_train, y_train)

coefficients = dict(zip(X.columns, model.coef_[0]))
print(coefficients)
```

```
{'Matchday (£m)': np.float64(-0.09190615923426007), 'Broadcasting (£m)': np.float64(-0.0029228448768425946), 'Commercial (£m
```

```
data = {
    'Club': ['AFC Bournemouth', 'Arsenal', 'Aston Villa', 'Brentford', 'Brighton & Hove Albion', 'Chelsea', 'Crystal Palace', 'E
    '2023 Revenue (£000)': [141004, 463570, 211950, 167596, 201169, 512467, 180066, 172155, 181188, 189684, 177326, 593836, 7182
    '2022 Revenue (£000)': [53857, 367554, 169123, 141277, 174461, 481278, 159999, 181007, 67092, 189207, 214590, 594271, 619082
}

df = pd.DataFrame(data)

X = df[['2022 Revenue (£000)']]
y = (df['2023 Revenue (£000)'] > df['2022 Revenue (£000)']).astype(int)

model = LogisticRegression()
model.fit(X,y)

y_pred = model.predict(X)

cm = confusion_matrix(y,y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title('Confusion Matrix: Logistic Regression Model Performance')
```

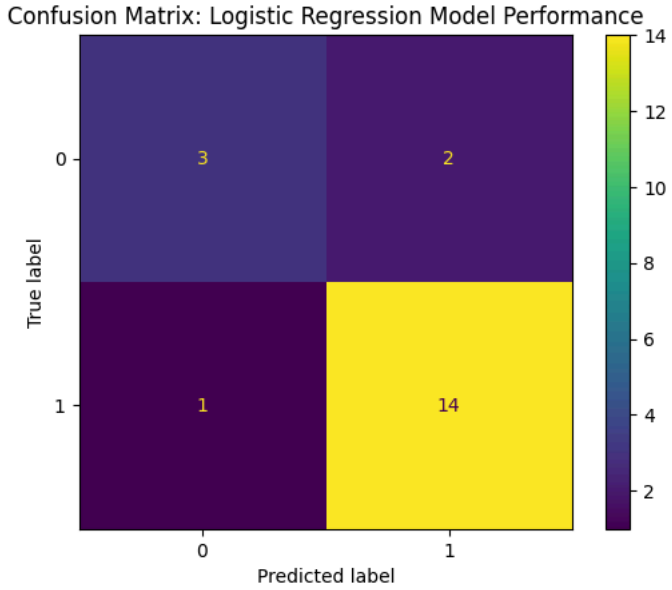
```
plt.show()
```

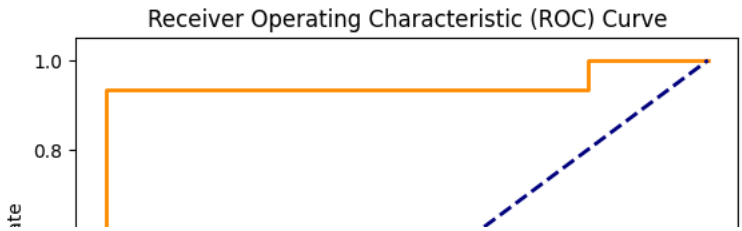



```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, cross_val_score, train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay, ro
import matplotlib.pyplot as plt

data = {
    'Club': ['AFC Bournemouth', 'Arsenal', 'Aston Villa', 'Brentford', 'Brighton & Hove Albion', 'Chelsea', 'Crystal Palace', 'Ev
    '2023 Revenue (£000)': [141004, 463570, 211950, 167596, 201169, 512467, 180066, 172155, 181188, 189684, 177326, 593836, 71822
    '2022 Revenue (£000)': [53857, 367554, 169123, 141277, 174461, 481278, 159999, 181007, 67092, 189207, 214590, 594271, 619082,
    '2023 Wage costs (£000)': [100109, 234766, 194236, 98800, 127563, 403962, 130626, 166094, 139064, 145857, 205780, 372881, 422
    '2023 Op result (£000)': [17940, 98241, -46977, 32838, 13626, -42540, 19925, -48051, 9318, -4491, -75174, 69416, 109810, 1410
    '2023 Net pft/(loss) on sale of ply reg. (£000)': [3912, 12189, 22462, 8071, 124422, 62861, 331, 47518, 8743, 73345, 74767, 3
}

df = pd.DataFrame(data)

X = df[['2022 Revenue (£000)', '2023 Wage costs (£000)', '2023 Op result (£000)', '2023 Net pft/(loss) on sale of ply reg. (£000)
y = (df['2023 Revenue (£000)'] > df['2022 Revenue (£000)']).astype(int)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

param_grid = {'C': [0.01, 0.1, 1, 10, 100]}
grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5)
grid_search.fit(X_scaled, y)

best_model = grid_search.best_estimator_

cv_scores = cross_val_score(best_model, X_scaled, y, cv=5)

y_pred = best_model.predict(X_scaled)

cm = confusion_matrix(y, y_pred)

accuracy = accuracy_score(y, y_pred)
precision = precision_score(y, y_pred)
recall = recall_score(y, y_pred)
f1 = f1_score(y, y_pred)

print(f'Confusion Matrix:\n{cm}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
print(f'Cross-validation scores: {cv_scores}')
print(f'Average cross-validation score: {cv_scores.mean()}')
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title('Confusion Matrix: Logistic Regression Model Performance')
plt.show()

fpr, tpr, thresholds = roc_curve(y, best_model.predict_proba(X_scaled)[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

X, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

model_rf = RandomForestClassifier(n_estimators=100, random_state=1)
model_rf.fit(X_train, y_train)

y_pred_rf = model_rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f'Random Forest Model Accuracy: {accuracy_rf}')


feature_importances = model_rf.feature_importances_
features = X.columns

plt.figure(figsize=(10, 6))
plt.barh(features, feature_importances, color='#00a398')
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Feature Importance in Random Forest Model')
plt.show()

cm_rf = confusion_matrix(y_test, y_pred_rf)
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf)
disp_rf.plot()
plt.title('Confusion Matrix: Random Forest Model Performance')
plt.show()
```

```
Confusion Matrix:
[[ 3  2]
 [ 1 14]]
Accuracy: 0.85
Precision: 0.875
Recall: 0.9333333333333333
F1 Score: 0.9032258064516129
Cross-validation scores: [1.   0.75 0.75 1.   0.75]
Average cross-validation score: 0.85
```






```
data = {
    'Club': ['AFC Bournemouth', 'Arsenal', 'Aston Villa', 'Brentford', 'Brighton & Hove Albion', 'Chelsea', 'Crystal Palace', 'E
    '2023 Wage costs (£000)': [100109, 234766, 194236, 98800, 127563, 403962, 130626, 166094, 139064, 145857, 205780, 372881, 42
    '2023 Op result (£000)': [17940, 98241, -46977, 32838, 13626, -42540, 19925, -48051, 9318, -4491, -75174, 69416, 109810, 141
}

df = pd.DataFrame(data)

X_wage = df[['2023 Wage costs (£000)']]
y_op_result = df['2023 Op result (£000)']

model_wage_op = LinearRegression()
model_wage_op.fit(X_wage, y_op_result)

y_op_result_pred = model_wage_op.predict(X_wage)

plt.scatter(X_wage, y_op_result, color='green')
plt.plot(X_wage, y_op_result_pred, color='orange')
plt.xlabel('2023 Wage costs (£000)')
```