In [33]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [34]:
```
hero = pd.read_csv('sample_101_Mlbb_Heroes.csv')

hero.head()
```

Out[34]:

	Name	Title	Voice_Line	Release_Date	Primary_Role	Secondary_Role	Lane
0	Aamon	Duke of Shards	It is better to be feared than loved, if you c...	2021-10-25	Assassin	NaN	Jungler
1	Akai	Panda Warrior	Now Akai enters the scene!	2016	Tank	Support	Roamer
2	Aldous	Soul Contractor	Primary Role	2018	Fighter	NaN	EXP Lane
3	Alice	Queen of Blood	Watch your back!	2016	Mage	Tank	EXP Lane
4	Alpha	Blade of Enmity	Test! Alpha is online.	2017	Fighter	NaN	EXP Lane

In [35]:
```
hero.info()
hero.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Name            100 non-null    object
 1   Title           100 non-null    object
 2   Voice_Line      100 non-null    object
 3   Release_Date    100 non-null    object
 4   Primary_Role    100 non-null    object
 5   Secondary_Role  28 non-null     object
 6   Lane            100 non-null    object
 7   Hp              100 non-null    int64
 8   Hp_Regen        100 non-null    float64
 9   Mana            100 non-null    int64
 10  Mana_Regen      100 non-null    int64
 11  Phy_Damage      100 non-null    int64
 12  Mag_Damage      100 non-null    int64
 13  Phy_Defence     100 non-null    int64
 14  Mag_Defence     100 non-null    int64
 15  Mov_Speed       100 non-null    int64
 16  Esport_Wins     100 non-null    int64
 17  Esport_Loss     100 non-null    int64
dtypes: float64(1), int64(10), object(7)
memory usage: 14.2+ KB
```

Out[35]:

	Hp	Hp_Regen	Mana	Mana_Regen	Phy_Damage	Mag_Damage
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.0
mean	2596.260000	7.730000	333.110000	14.850000	116.480000	0.0
std	149.658306	1.654226	215.159508	24.292151	9.886712	0.0
min	2043.000000	3.800000	0.000000	0.000000	90.000000	0.0
25%	2501.000000	6.800000	75.000000	2.250000	110.000000	0.0
50%	2575.500000	7.300000	432.000000	15.000000	117.000000	0.0
75%	2711.250000	8.250000	481.500000	18.000000	123.000000	0.0
max	2909.000000	18.400000	750.000000	240.000000	140.000000	0.0

In [36]:
```
hero = hero.drop(
    columns=["Secondary_Role", "Title", "Voice_Line", "Release_Date"],
    errors="ignore"
)
```

In [37]:
```
hero.head()
```

Out[37]:

	Name	Primary_Role	Lane	Hp	Hp_Regen	Mana	Mana_Regen	Phy_Damage
0	Aamon	Assassin	Jungler	2614	8.0	455	21	115
1	Akai	Tank	Roamer	2769	8.4	422	12	115
2	Aldous	Fighter	EXP Lane	2718	9.8	405	18	129
3	Alice	Mage	EXP Lane	2573	7.2	493	18	114
4	Alpha	Fighter	EXP Lane	2646	7.8	453	31	121

In [38]:
```
hero.isnull().sum()
```

Out[38]:
```
Name            0
Primary_Role    0
Lane            0
Hp              0
Hp_Regen        0
Mana            0
Mana_Regen      0
Phy_Damage      0
Mag_Damage      0
Phy_Defence     0
Mag_Defence     0
Mov_Speed       0
Esport_Wins     0
Esport_Loss     0
dtype: int64
```

In [39]:
```
hero.dtypes
```

Out[39]:
```
Name            object
Primary_Role    object
Lane            object
Hp               int64
Hp_Regen       float64
Mana             int64
Mana_Regen       int64
Phy_Damage       int64
Mag_Damage       int64
Phy_Defence      int64
Mag_Defence      int64
Mov_Speed        int64
Esport_Wins      int64
Esport_Loss      int64
dtype: object
```

In [40]:
```
hero['Total_damage'] = hero['Phy_Damage'] + hero['Mag_Damage']
```

In [41]:
```
hero['Total_defence'] = hero['Phy_Defence'] + hero['Mag_Defence']
```

In [42]:
```
hero['Win_rate'] = hero["Esport_Wins"] / (
    hero["Esport_Wins"] + hero["Esport_Loss"]
)
```

In [43]:
```
hero['Win_rate'].describe()
```

Out[43]:
```
count    100.000000
mean       0.477461
std        0.075023
min        0.000000
25%        0.462852
50%        0.497482
75%        0.515519
max        0.578571
Name: Win_rate, dtype: float64
```

In [44]:
```
hero.duplicated().sum()
```

Out[44]:
```
np.int64(0)
```

In [45]:
```
hero.head()
```

Out[45]:

	Name	Primary_Role	Lane	Hp	Hp_Regen	Mana	Mana_Regen	Phy_Damage
0	Aamon	Assassin	Jungler	2614	8.0	455	21	115
1	Akai	Tank	Roamer	2769	8.4	422	12	115
2	Aldous	Fighter	EXP Lane	2718	9.8	405	18	129
3	Alice	Mage	EXP Lane	2573	7.2	493	18	114
4	Alpha	Fighter	EXP Lane	2646	7.8	453	31	121

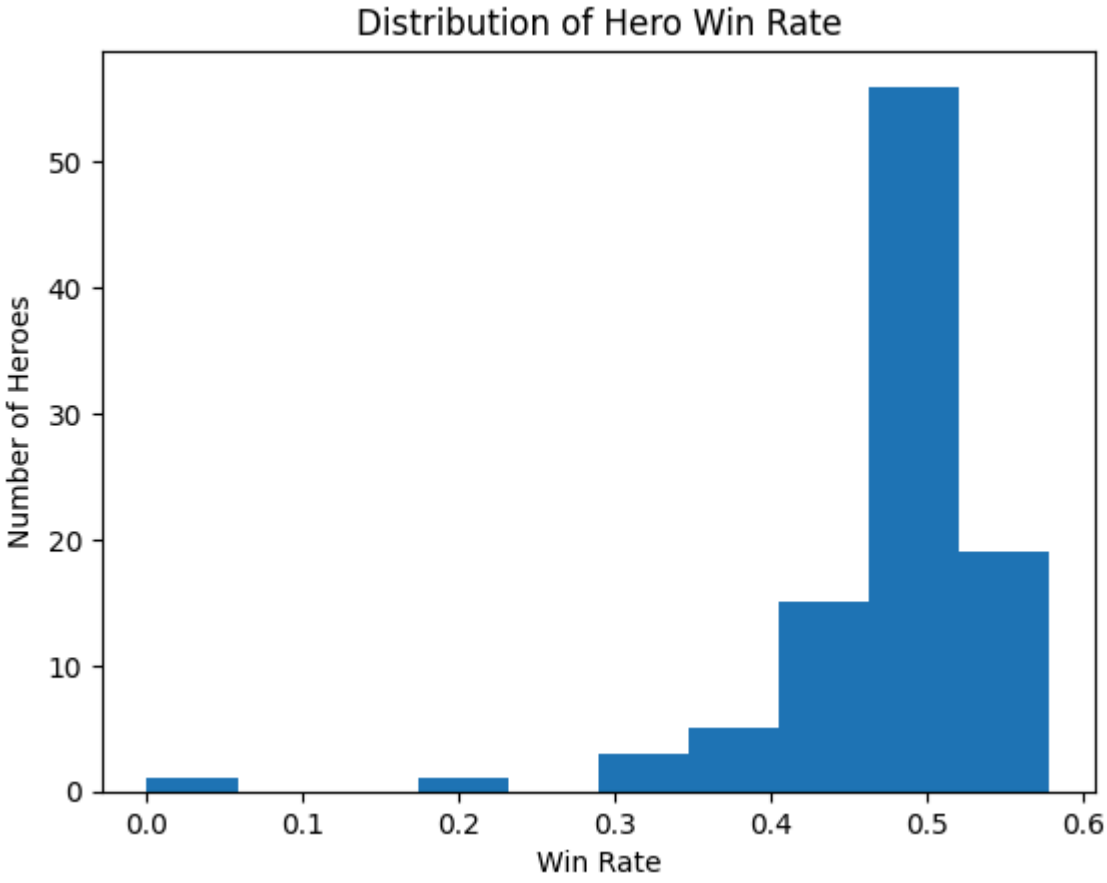
In [46]:
```
hero['Overall_score'] = (hero['Total_damage'] + hero['Total_defence'] + hero['Mo
```

In [47]:
```
hero.head()
```

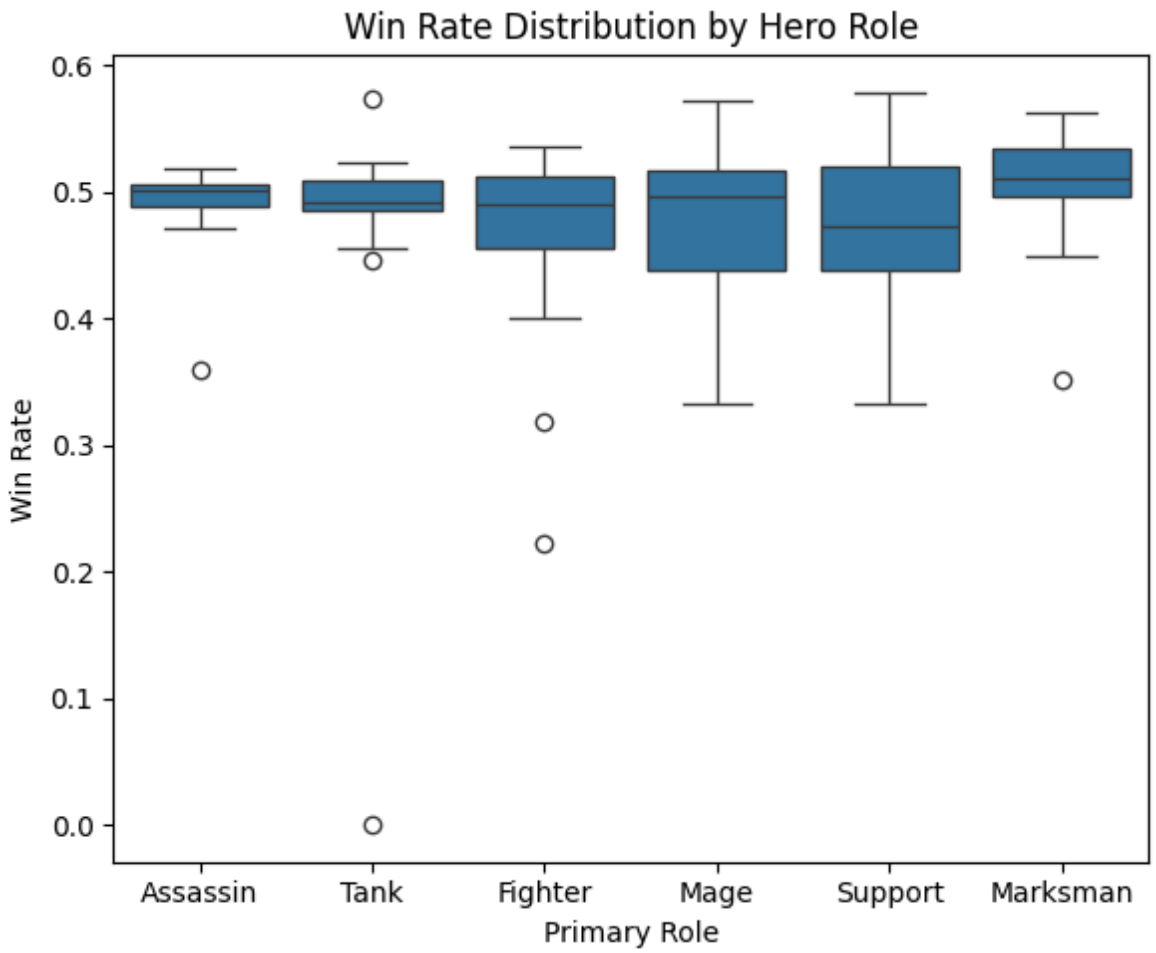
Out[47]:

	Name	Primary_Role	Lane	Hp	Hp_Regen	Mana	Mana_Regen	Phy_Damage
0	Aamon	Assassin	Jungler	2614	8.0	455	21	115
1	Akai	Tank	Roamer	2769	8.4	422	12	115
2	Aldous	Fighter	EXP Lane	2718	9.8	405	18	129
3	Alice	Mage	EXP Lane	2573	7.2	493	18	114
4	Alpha	Fighter	EXP Lane	2646	7.8	453	31	121

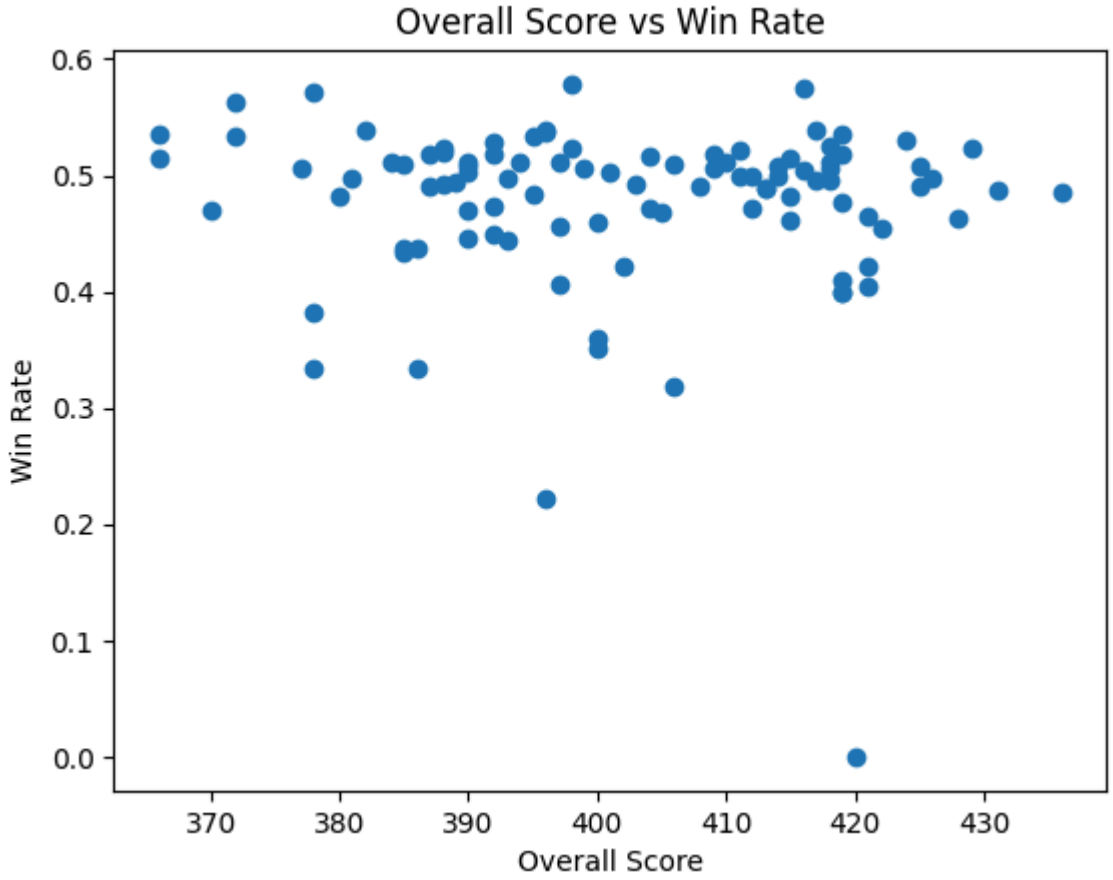
In [48]:
```
plt.figure()
plt.hist(hero["Win_rate"], bins=10)
plt.xlabel("Win Rate")
plt.ylabel("Number of Heroes")
plt.title("Distribution of Hero Win Rate")
plt.show()
```



In [63]:
```
plt.figure(figsize=(6,5))
sns.boxplot(x="Primary_Role",y="Win_rate",data=hero)
plt.xlabel("Primary Role")
plt.ylabel("Win Rate")
plt.title("Win Rate Distribution by Hero Role")
plt.tight_layout()
plt.show()
```

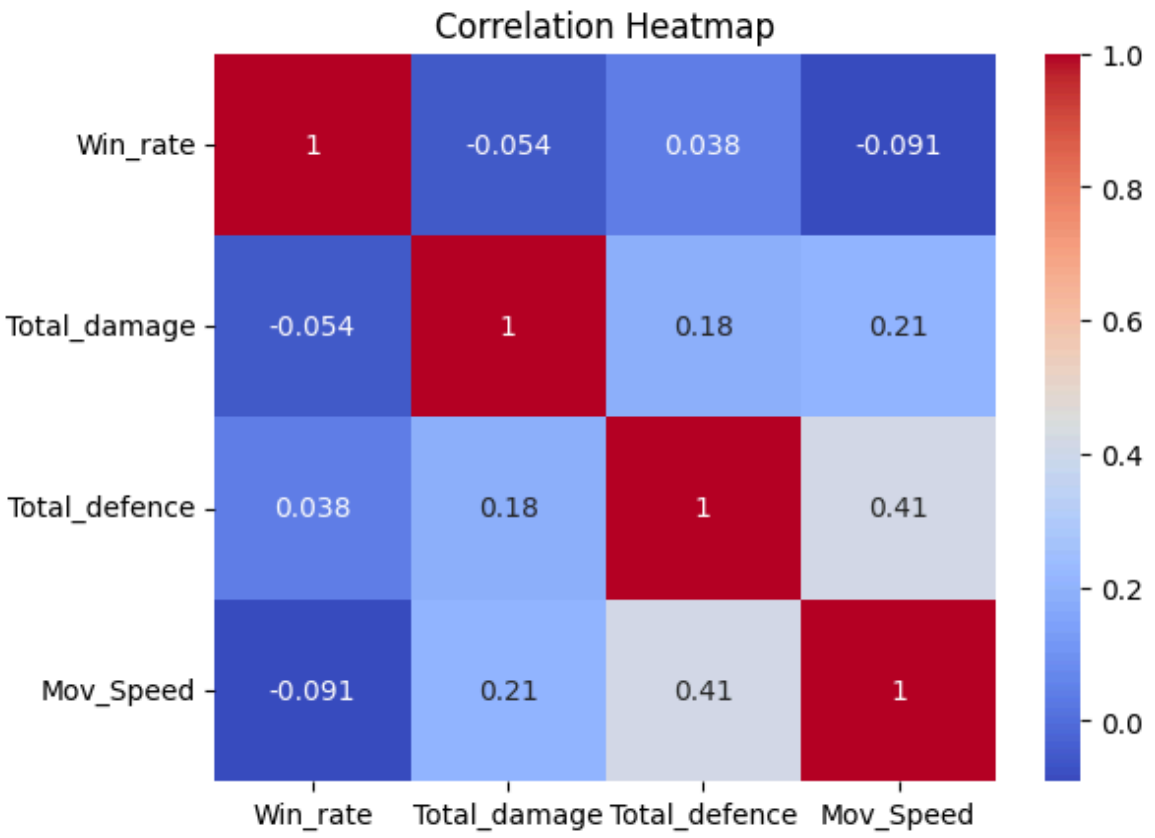


In [50]:
```
plt.figure()
plt.scatter(hero["Overall_score"], hero["Win_rate"])
plt.xlabel("Overall Score")
plt.ylabel("Win Rate")
plt.title("Overall Score vs Win Rate")
plt.show()
```



In [56]:
```
corr = hero[["Win_rate", "Total_damage", "Total_defence", "Mov_Speed"]].corr()
```

In [59]:
```
plt.figure()
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



In [21]:
```
X = hero[['Overall_score']]
y = hero['Win_rate']
```

In [22]:
```
from sklearn.model_selection import train_test_split
```

In [23]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_sta
```

In [24]:
```
from sklearn.linear_model import LinearRegression
```

In [25]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[25]:

▾ LinearRegression

▶ Parameters

In [29]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

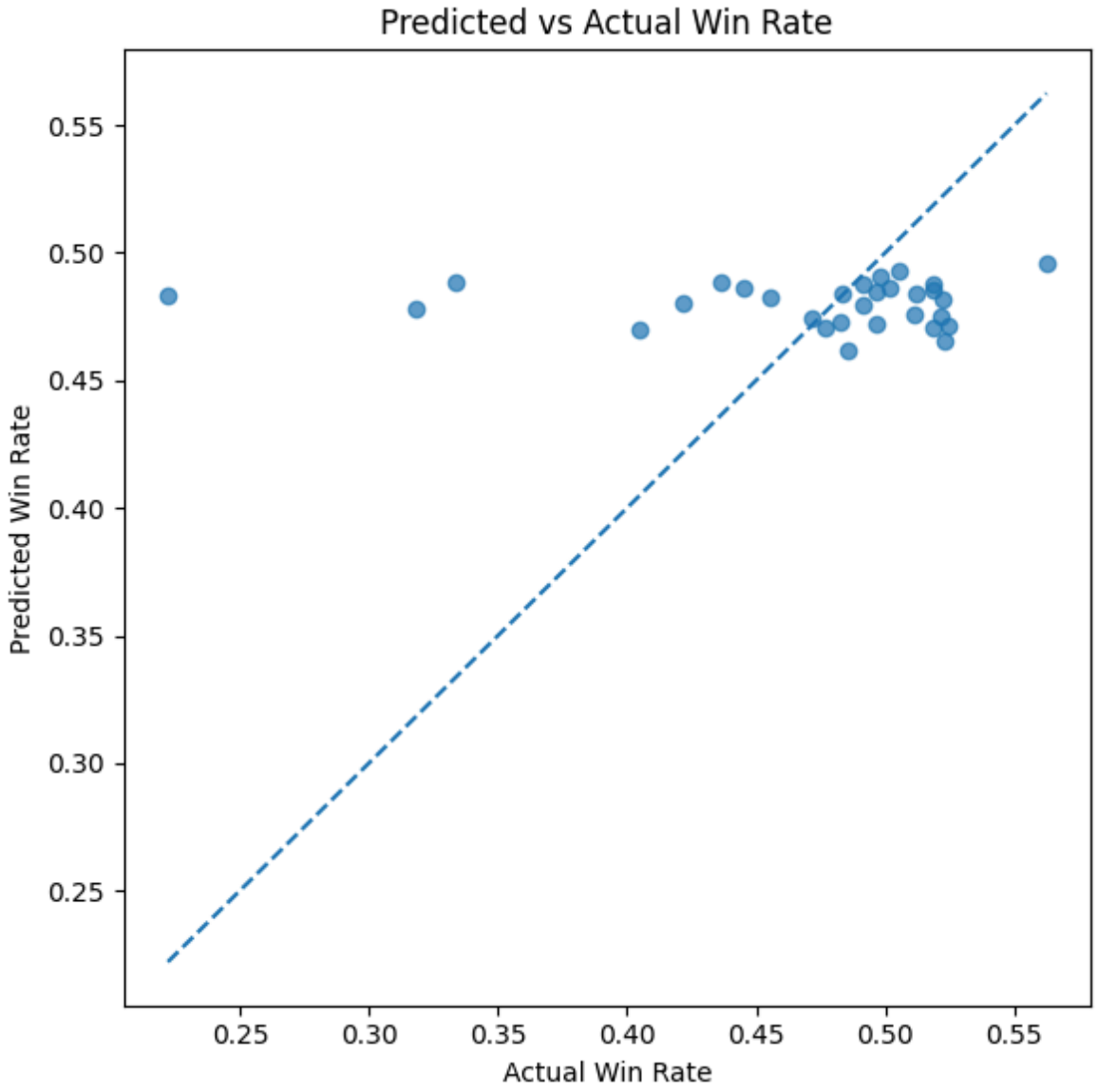
In [30]:
```
y_pred = model.predict(X_test)
```

In [31]:
```
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print("MAE:", mae)
print("RMSE:", rmse)
print("R2:", r2)
```

```
MAE: 0.04624414232006068
RMSE: 0.07142764009669994
R2: -0.03492917418200525
```

In [67]:
```
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()],
         linestyle="--")
plt.xlabel("Actual Win Rate")
plt.ylabel("Predicted Win Rate")
plt.title("Predicted vs Actual Win Rate")
plt.tight_layout()
plt.show()
```



In [36]:
```
import joblib
joblib.dump(model, "draft.joblib")
print("Model saved as draft.joblib")
```

```
Model saved as draft.joblib
```

In [ ]: