

```
In [140... import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
```

```
In [141... basketball_df = pd.read_csv('final_project.csv')
print(basketball_df.columns)
basketball_df.head(10)
```

```
Index(['TEAM', 'YEAR', 'Unnamed: 0', 'Rk', 'Conf', 'W-L', 'NetRtg', 'ORtg',
      'DRtg', 'AdjT', 'Luck', 'NetRtg.1', 'ORtg.1', 'DRtg.1', 'NetRtg.2',
      'CONF', 'G', 'W', 'ADJOE', 'ADJDE', 'BARTHAG', 'EFG_0', 'EFG_D', 'TO
R',
      'TORD', 'ORB', 'DRB', 'FTR', 'FTRD', '2P_0', '2P_D', '3P_0', '3P_D',
      'ADJ_T', 'WAB', 'POSTSEASON', 'SEED'],
      dtype='object')
```

Out [141]...

	TEAM	YEAR	Unnamed: 0	Rk	Conf	W-L	NetRtg	ORtg	DRtg	AdjT	...	FTR	F
0	Air Force	2013	98.0	99.0	MWC	18-14	6.74	111.7	104.9	63.2	...	30.8	
1	Air Force	2014	251.0	252.0	MWC	12-18	-7.00	99.6	106.6	63.6	...	37.1	
2	Air Force	2015	203.0	204.0	MWC	14-17	-3.01	108.1	111.2	61.7	...	30.8	
3	Air Force	2016	241.0	242.0	MWC	14-18	-6.92	99.1	106.0	67.0	...	39.6	
4	Air Force	2017	221.0	222.0	MWC	12-21	-5.22	104.3	109.5	67.1	...	37.4	
5	Air Force	2018	236.0	237.0	MWC	12-19	-6.25	100.5	106.8	66.0	...	32.0	
6	Air Force	2019	242.0	243.0	MWC	14-18	-7.06	100.6	107.7	66.6	...	28.3	
7	Air Force	2021	331.0	332.0	MWC	5-20	-16.51	94.4	110.9	62.8	...	30.9	
8	Air Force	2022	268.0	269.0	MWC	11-18	-8.48	95.5	104.0	63.2	...	26.8	
9	Air Force	2023	146.0	147.0	MWC	14-18	1.25	105.8	104.5	63.0	...	31.2	

10 rows x 37 columns

In [142]...

```
# Readjusting dataset based on VIF and NA
basketball_df = basketball_df.drop(['NetRtg.1', 'ORtg.1', 'DRtg.1', 'NetRtg.1'])
basketball_df = basketball_df.dropna()

# calculating win percent
basketball_df['Win Percent'] = (basketball_df['W'] / basketball_df['G']) * 100

# Creating target and feature datasets
X = basketball_df.drop('Win Percent', axis = 1)
y = basketball_df['Win Percent']
```

In [143]...

```
# One Hot Encoding Conference -- now have conference encoded and other features
X_encoded = pd.get_dummies(X, columns = ['CONF'], drop_first=True)

# Removing Conference first to check VIF
X = X_encoded.drop(['CONF'], axis = 1)

# Checking Multicollinearity
vif_data = pd.DataFrame()
vif_data['Variable'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

	Variable	VIF
0	Unnamed: 0	2.122017e+08
1	Rk	2.135816e+08
2	NetRtg	8.396812e+04
3	ORtg	3.049530e+04
4	DRtg	2.505789e+04
5	AdjT	1.190834e+00
6	Luck	2.516317e+00
7	G	5.153025e+00
8	W	2.427287e+01
9	ADJOE	1.435666e+02
10	ADJDE	9.036510e+01
11	BARTHAG	2.051573e+02
12	EFG_0	9.628400e+00
13	EFG_D	7.984067e+00
14	TOR	3.876166e+00
15	TORD	4.348699e+00
16	ORB	3.604183e+00
17	DRB	3.349190e+00
18	FTR	1.514659e+00
19	FTRD	1.903236e+00
20	3P_0	2.530351e+00
21	WAB	3.541024e+01

```
In [148... # Finalize X and y feature selection off of VIF

X = X[['Luck', 'EFG_D', 'EFG_0', 'TOR', 'TORD', 'ORB', 'DRB', 'FTR', 'FTRD',

# Train and test model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, ra
```

```
In [153... # Fitting and predicting the model
model_bball = LinearRegression()
model_bball.fit(X_train, y_train)
y_pred = model_bball.predict(X_test)

# Calculating MSE and R^2
rmse = np.sqrt(mean_squared_error(y_test, lasso_pred))
r2 = r2_score(y_test, y_pred)

#Must check for linearity (checked), homoskedacity, and normality

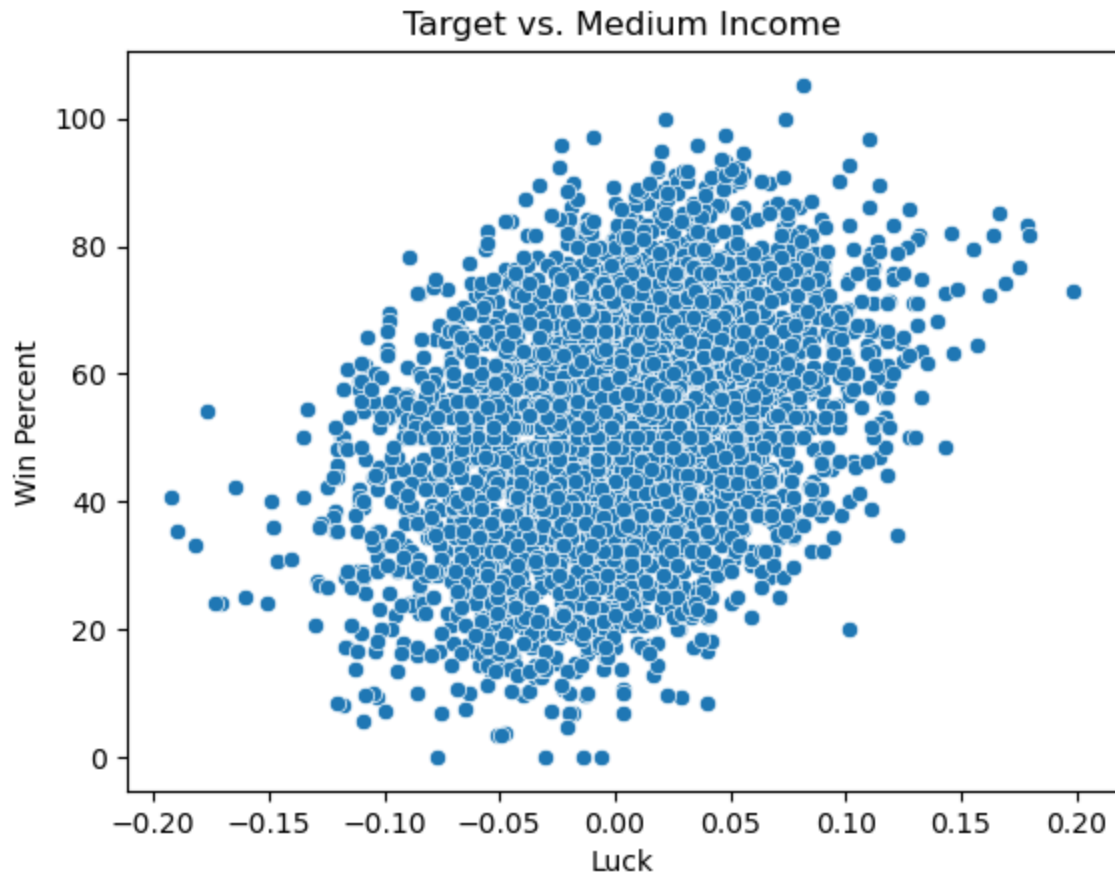
# residuals vs fitted values plot
residuals = y_test - y_pred

sns.scatterplot(x = y_pred, y = residuals)
plt.axhline(y = 0, color='red', linestyle='--')
plt.title("Residuals vs Fitted Values")
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.show();

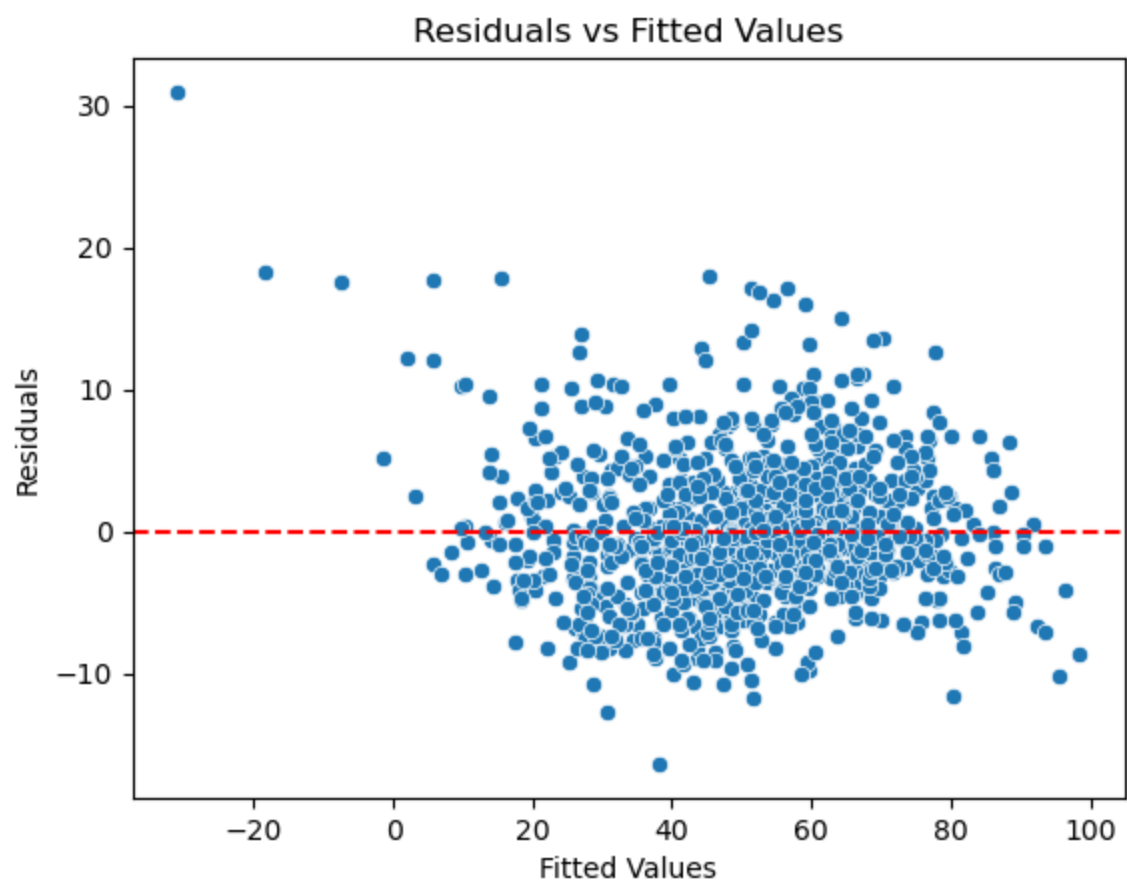
# Q-Q Plot
sm.qqplot(residuals, line='s')
plt.title("Q-Q Plot of Residuals")
plt.show();
```

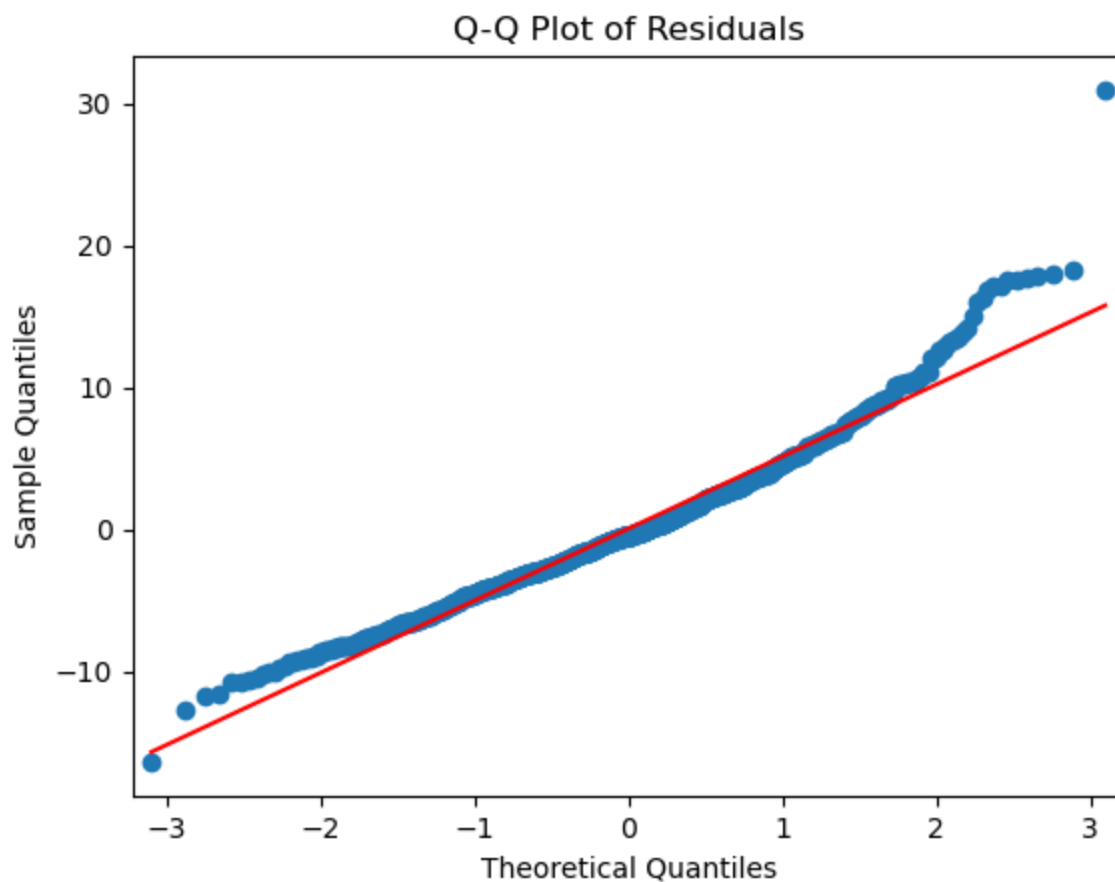
```
#print(model_bball.intercept_)  
#print(model_bball.coef_)  
rmse, r2
```











Out[153... (5.396914138220759, 0.9220616249379828)

```
In [154... X_train_scaled = X_train
X_test_scaled = X_test

ridge = Ridge(alpha = 1.0)
ridge.fit(X_train_scaled, y_train)

# Predict and evaluate Ridge model
ridge_pred = ridge.predict(X_test_scaled)
ridge_r2 = r2_score(y_test, ridge_pred)
ridge_rmse = np.sqrt(mean_squared_error(y_test, ridge_pred))

# Coefficients and Intercept
ridge_coefficients = ridge.coef_
ridge_intercept = ridge.intercept_

print("Ridge Regression Model:")
print(f'RMSE: {ridge_rmse:.4f}')
print(f'R^2: {ridge_r2:.4f}')
```

Ridge Regression Model:
 RMSE: 5.1222
 R^2: 0.9209

```
In [157... X_train_scaled = X_train
X_test_scaled = X_test

lasso = Lasso(alpha=0.001)
```

```

lasso.fit(X_train_scaled, y_train)

# Predict and evaluate Lasso model
lasso_pred = lasso.predict(X_test_scaled)
lasso_r2 = r2_score(y_test, lasso_pred)
lasso_rmse = np.sqrt(mean_squared_error(y_test, lasso_pred))

# Coefficients and Intercept
lasso_coefficients = lasso.coef_
lasso_intercept = lasso.intercept_

print("Lasso Regression Model:")
print(f"R^2: {lasso_r2:.4f}")
print(f"RMSE: {lasso_rmse:.4f}")

```

Lasso Regression Model:

R²: 0.9221

RMSE: 5.0846

```

In [156... X_train_scaled = X_train
X_test_scaled = X_test

elastic_net = ElasticNet()

# Define the parameter grid to search over
param_grid = {
    'alpha': [0.001, 0.01, 0.1, 1, 10, 100], # Regularization strength
    'l1_ratio': [0.1, 0.2, 0.5, 0.7, 0.9, 1] # Balance between Lasso and Ridge
}

# Set up GridSearchCV with 3-fold cross-validation
grid_search = GridSearchCV(estimator=elastic_net, param_grid=param_grid,
                           cv = 5,
                           scoring='neg_root_mean_squared_error')

# Fit the model using GridSearchCV
grid_search.fit(X_train_scaled, y_train)

# Get the best model and hyperparameters
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_

# Predict with the best model
y_pred = best_model.predict(X_test_scaled)

# Evaluate the model
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Coefficients and Intercept
elastic_coefficients = best_model.coef_
elastic_intercept = best_model.intercept_

# Print the results
print(f"Best Hyperparameters: {best_params}")

```

```
print(f"R2 on Test Set: {r2:.4f}")  
print(f"RMSE on Test Set: {rmse:.4f}")
```

Best Hyperparameters: {'alpha': 0.001, 'l1_ratio': 1}

R² on Test Set: 0.9221

RMSE on Test Set: 5.0846