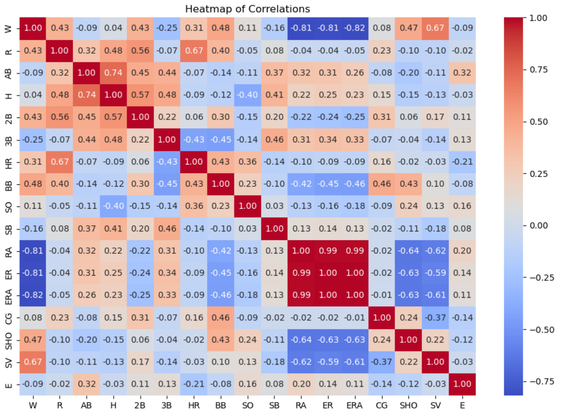
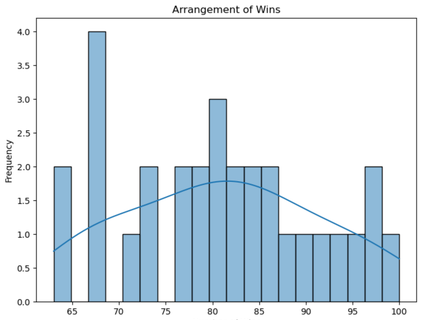
In [1]:
```
# Step 1: Import required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
import joblib

# Step 2: Load the dataset
link = "https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/baseball.csv"
baseball = pd.read_csv(link)

# Step 3: Exploratory Data Analysis (EDA)
# Display the first few rows of the dataset
print(baseball.head())

# Get summary statistics of the numerical features
print(baseball.describe())

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(baseball.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Heatmap of Correlations")
plt.show()

# Visualize the distribution of the target variable 'W'
plt.figure(figsize=(8, 6))
sns.histplot(baseball['W'], bins=20, kde=True)
plt.title('Arrangement of Wins')
plt.xlabel('Number of Wins')
plt.ylabel('Frequency')
plt.show()

# Step 4: Preprocessing and Feature Engineering
# For simplicity, let's drop any rows with missing values
df = baseball.dropna()

# Define features and target variable
X = baseball.drop(columns=['W'])
y = baseball['W']

# Step 5: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Standardize numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 7: Build and Test Multiple Models
# Model 1: Linear Regression
model = LinearRegression()
model.fit(X_train_scaled, y_train)
y_pred= model.predict(X_test_scaled)

# Model 2: Random Forest
model_2 = RandomForestRegressor()
model_2.fit(X_train_scaled, y_train)
y_pred2= model_2.predict(X_test_scaled)

# Step 8: Evaluate Models
def evaluate_model(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    return mse, r2

mse, r2_l = evaluate_model(y_test, y_pred)
mse2, r2_r= evaluate_model(y_test, y_pred2)

print(f"Linear Regression - MSE: {mse}, R2: {r2_l}")
print(f"Random Forest - MSE: {mse2}, R2: {r2_r}")

# Step 9: Model Selection
selected_model = model if r2_l > r2_r else model_2
print("Selected Model:", selected_model)

# Step 10: Hyperparameter Tuning
paramgrid = {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]}
gridsearch = GridSearchCV(RandomForestRegressor(), paramgrid, cv=5)
gridsearch.fit(X_train_scaled, y_train)

best_rf_model = gridsearch.best_estimator_
print("Best Random Forest Model:", best_rf_model)

# Step 11: Save the Best Model
joblib.dump(best_rf_model, 'best_model.pkl')
```

```
    W    R    AB    H   2B  3B   HR   BB    SO   SB   RA   ER  ERA  CG  SHO  \
0  95  724  5575  1497  300  42  139  383   973  104  641  601  3.73   2    8
1  83  696  5467  1349  277  44  156  439  1264   70  700  653  4.07   2   12
2  81  669  5439  1395  303  29  141  533  1157   86  640  584  3.67  11   10
3  76  622  5533  1381  260  27  136  404  1231   68  701  643  3.98   7    9
4  74  689  5605  1515  289  49  151  455  1259   83  803  746  4.64   7   12

   SV    E
0  56   88
1  45   86
2  38   79
3  37  101
4  35   86
                W            R           AB           H           2B  \
count   30.000000    30.000000    30.000000    30.000000    30.000000
mean    80.966667   688.233333  5516.266667  1403.533333   274.733333
std     10.453455    58.761754    70.467372    57.140923    18.095405
min     63.000000   573.000000  5385.000000  1324.000000   236.000000
25%     74.000000   651.250000  5464.000000  1363.000000   262.250000
50%     81.000000   689.000000  5510.000000  1382.500000   275.500000
75%     87.750000   718.250000  5570.000000  1451.500000   288.750000
max    100.000000   891.000000  5649.000000  1515.000000   308.000000

               3B          HR          BB          SO          SB          RA  \
count   30.000000   30.000000   30.000000   30.000000   30.000000   30.000000
mean    31.300000  163.633333  469.100000  1248.20000   83.500000  688.233333
std     10.452355   31.823309   57.053725   103.75947   22.815225   72.108005
min     13.000000  100.000000  375.000000   973.00000   44.000000  525.000000
25%     23.000000  140.250000  428.250000  1157.50000   69.000000  636.250000
50%     31.000000  158.500000  473.000000  1261.50000   83.500000  695.500000
75%     39.000000  177.000000  501.250000  1311.50000   96.500000  732.500000
max     49.000000  232.000000  570.000000  1518.00000  134.000000  844.000000

               ER         ERA          CG         SHO          SV           E
count   30.000000   30.000000   30.000000   30.000000   30.000000   30.000000
mean   635.833333    3.956333    3.466667   11.300000   43.066667   94.333333
std     70.140786    0.454089    2.763473    4.120177    7.869335   13.958889
min    478.000000    2.940000    0.000000    4.000000   28.000000   75.000000
25%    587.250000    3.682500    1.000000    9.000000   37.250000   86.000000
50%    644.500000    4.025000    3.000000   12.000000   42.000000   91.000000
75%    679.250000    4.220000    5.750000   13.000000   46.750000   96.750000
max    799.000000    5.040000   11.000000   21.000000   62.000000  126.000000
```






```
Linear Regression - MSE: 27.94303250666786, R2: 0.7876400316149371
Random Forest - MSE: 54.67500000000003, R2: 0.5844838505383152
Selected Model: LinearRegression()
Best Random Forest Model: RandomForestRegressor(max_depth=20)
['best_model.pkl']
```

Out[1]:

Baseball Dataset Regression Modeling Documentation

Introduction: This Python code focuses on performing regression modeling on a baseball dataset, target to analysis the number of wins (W) based on various features. The analysis involves multiple steps, including data loading, exploratory data analysis (EDA), preprocessing, model building, evaluation, and hyperparameter tuning.

Libraries: pandas and numpy: Utilized for data manipulation and numerical operations. seaborn and matplotlib.pyplot: Employed for data visualization. scikit-learn (sklearn): Utilized for machine learning tasks, including preprocessing, modeling, and evaluation. joblib: Used for saving the best model for future use. Steps:

1. Import Required Libraries: Imported essential libraries for data manipulation, visualization, machine learning, and model persistence.
2. Load the Dataset: Fetched the baseball dataset from a specified URL using pandas.
3. Exploratory Data Analysis (EDA): Arrange EDA to understand the dataset by displaying the initial rows, obtaining summary statistics, and visualizing correlations and the distribution of the target variable (W).
4. Preprocessing and Feature Engineering: Managed missing values by dropping rows, and defined features (X) and the target variable (y).
5. Split the Data: Separated the dataset into training and testing sets using the train_test_split function.
6. Standardize Numerical Features: Used StandardScaler to standardize numerical features for better model performance.
7. Build and Test Multiple Models: Constructed and evaluated two models - Linear Regression and Random Forest Regression.
8. Evaluate Models: find mean squared error (MSE) and R-squared (R2) to assess the performance of both models.
9. Model Selection: Selected the model with the higher R2 value as the preferred model for further analysis.
10. Hyperparameter Tuning: Use GridSearchCV to perform hyperparameter tuning on the Random Forest model.
11. Save the Best Model: Stored the best-performing Random Forest model using joblib for future use.

Conclusion: The code provides a comprehensive approach to regression modeling on the baseball dataset, covering key data science tasks from data exploration to model selection and tuning. The selected best model is saved for potential deployment or further analysis.