Noah Anderson

Module 5

July 5th, 2024

Question 1

In [ ]:
```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import math
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from scipy import stats
```

a.

In [ ]:
```
# Read in baseball players data
players = pd.read_csv("hall_of_fame.csv")

# Create singles column
players['singles'] = players['hits'] - players['doubles'] - players['triples'] - players['HR']

# Create data frame selecting only hitting related data
hitting_df = (players[['runs', 'AB', 'singles', 'doubles', 'triples', 'HR', 'BB', 'SO']]
              .dropna()) # Drop NA's
```

b.

In [ ]:
```
# Fit linear model for runs using hitting variables
model1 = smf.ols('runs ~ singles + doubles + triples + HR + BB + SO',
                 data = hitting_df).fit()
```

```python
# Summarize model
model1.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	runs	**R-squared:**	0.949
Model:	OLS	**Adj. R-squared:**	0.949
Method:	Least Squares	**F-statistic:**	4098.
Date:	Tue, 25 Jun 2024	**Prob (F-statistic):**	0.00
Time:	10:10:28	**Log-Likelihood:**	-7739.6
No. Observations:	1320	**AIC:**	1.549e+04
Df Residuals:	1313	**BIC:**	1.553e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-24.6262	6.536	-3.768	0.000	-37.449	-11.803
singles	0.3875	0.013	28.858	0.000	0.361	0.414
doubles	-0.0375	0.058	-0.649	0.517	-0.151	0.076
triples	2.5550	0.097	26.455	0.000	2.366	2.744
HR	0.6610	0.038	17.206	0.000	0.586	0.736
BB	0.3230	0.014	23.216	0.000	0.296	0.350
SO	-0.0417	0.013	-3.329	0.001	-0.066	-0.017

Omnibus:	263.662	**Durbin-Watson:**	2.002
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1342.745
Skew:	0.834	**Prob(JB):**	2.67e-292
Kurtosis:	7.651	**Cond. No.**	3.36e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.36e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

c.

The R squared stasitic explains the percentage of variation explained by the model. The adjusted R squared for this model is .949 which is considerably high indicating that this likely fits the model well.

d.

The relationship of the doubles coeffecient being slightly negative is counterintiutive. One explanation is that it is close to zero, but slightly negative with a non-significant p-value of .517 implying that there is low confidence in this relationship with runs. Additionally it feels wrong to include home runs since in theory this should be a 1-to-1 ratio since scoring a homerun results in run by definition. The relationship of .6610 is less than I would expect.

Question 2

a.

At-bats are closely related to other hitting statistics because more at-bats provide more opportunities to accumulate these statistics. For instance, an average player who has played five times longer than an excellent player will naturally have more at-bats, resulting in a higher accumulation of hitting statistics.

b.

In [ ]:
```
# Set options so that the entire correlation matrix is printed
pd.set_option('display.max_columns', None)
```

```
# Show correlation matrix for hitting data
hitting_df.corr()
```

Out[ ]:

	runs	AB	singles	doubles	triples	HR	BB	SO
runs	1.000000	0.930846	0.921744	0.899165	0.785519	0.562797	0.827109	0.059733
AB	0.930846	1.000000	0.966941	0.925077	0.707119	0.560482	0.776491	0.122526
singles	0.921744	0.966941	1.000000	0.896605	0.767146	0.396568	0.704995	0.069465
doubles	0.899165	0.925077	0.896605	1.000000	0.712002	0.585208	0.751428	0.125205
triples	0.785519	0.707119	0.767146	0.712002	1.000000	0.160062	0.493767	-0.010294
HR	0.562797	0.560482	0.396568	0.585208	0.160062	1.000000	0.663796	0.125668
BB	0.827109	0.776491	0.704995	0.751428	0.493767	0.663796	1.000000	0.120935
SO	0.059733	0.122526	0.069465	0.125205	-0.010294	0.125668	0.120935	1.000000

At-bats are highly correlated with positive hitting statistics, with the exception of strikeouts. The correlation with home runs is the weakest among these, with a value of 0.56. Singles are highly correlated at .97 which makes sense considering it is the most achievable out of the types of hits (singles, doubles, triples, and home runs).

C.

In [ ]:

```
# Fit model for runs using only at-bats as a predictor
model2 = smf.ols('runs ~ AB', data = hitting_df).fit()

# Show summary of at-bats model
model2.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	runs	**R-squared:**	0.866
Model:	OLS	**Adj. R-squared:**	0.866
Method:	Least Squares	**F-statistic:**	8553.
Date:	Tue, 25 Jun 2024	**Prob (F-statistic):**	0.00
Time:	10:10:29	**Log-Likelihood:**	-8378.9
No. Observations:	1320	**AIC:**	1.676e+04
Df Residuals:	1318	**BIC:**	1.677e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-122.9048	9.047	-13.585	0.000	-140.653	-105.156
AB	0.1672	0.002	92.481	0.000	0.164	0.171

Omnibus:	305.818	**Durbin-Watson:**	2.068
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1220.021
Skew:	1.061	**Prob(JB):**	1.19e-265
Kurtosis:	7.205	**Cond. No.**	1.19e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.19e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

The R-squared value for this model is 0.866, indicating that it is quite effective at explaining the variance in runs based on at-bats. This value is only 0.093 lower than the R-squared for the first model. However, the apparent predictive power of the more inclusive model may be misleading due to multicollinearity, which will be explored in the next question.

Question 3

a.

In [ ]:
```
# Create per-at-bats data frame
hitting_rates_df = (hitting_df
                    .drop(columns = 'AB')
                    .div(hitting_df['AB'], axis = 0))
```

b.

In [ ]:
```
# Fit per-at-bats model
model3 = smf.ols('runs ~ singles + doubles + triples + HR + BB + SO',
                 data = hitting_rates_df).fit()

# Show model summary
model3.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	runs	**R-squared:**	0.643
Model:	OLS	**Adj. R-squared:**	0.641
Method:	Least Squares	**F-statistic:**	393.3
Date:	Tue, 25 Jun 2024	**Prob (F-statistic):**	4.73e-289
Time:	10:10:29	**Log-Likelihood:**	3424.2
No. Observations:	1320	**AIC:**	-6834.
Df Residuals:	1313	**BIC:**	-6798.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0358	0.006	-5.587	0.000	-0.048	-0.023
singles	0.4840	0.030	16.133	0.000	0.425	0.543
doubles	0.0941	0.063	1.494	0.135	-0.029	0.218
triples	2.7476	0.105	26.142	0.000	2.541	2.954
HR	0.7909	0.049	16.245	0.000	0.695	0.886
BB	0.3036	0.015	20.124	0.000	0.274	0.333
SO	-0.0217	0.006	-3.553	0.000	-0.034	-0.010

Omnibus:	102.098	**Durbin-Watson:**	1.967
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	197.493
Skew:	0.515	**Prob(JB):**	1.30e-43
Kurtosis:	4.590	**Cond. No.**	223.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

c.

The adjusted R-squared is considerably less than the non-rate based model at .641. This could be more reasonable than an R-squared of .959 given that a player's ability to score is depedent on the team around them as well.

Question 4

a.

In [ ]:
```
# Define predictor data frame
X = hitting_rates_df.drop(columns = 'runs')

# Define outcome data frame
Y = hitting_rates_df['runs']

# Split hitting rates into training and testing data
X_train, X_test, Y_train, Y_test = \
  train_test_split(X, Y, test_size = 0.30, random_state =328)

# fit model
fit = LinearRegression().fit(X_train, Y_train)

# Predict test data
predicted_y = fit.predict(X_test)

# Calculate mse
mse = mean_squared_error(Y_test, predicted_y)

# calculate rmse
rmse = math.sqrt(mse)

# Print model metrics
```

```
print("MSE =", round(mse, 3))
print("RMSE =", round(rmse, 3))
```

```
MSE = 0.0
RMSE = 0.018
```

b.

In [ ]:
```
# Read in 2022 data
players_2022 = pd.read_csv("players_2022.csv")

hitting_2022_df = players_2022.drop(columns = 'playerID')

X22 = hitting_2022_df.drop(columns = 'runs')
Y22 = hitting_2022_df['runs']

# Split 2022 hitting rates into training and testing data
X22_train, X22_test, Y22_train, Y22_test = \
    train_test_split(X22, Y22, test_size = .30, random_state = 234)

# Fit model on 2022 training data
fit22 = LinearRegression().fit(X22_train, Y22_train)

# Predict test data
predicted_y22 = fit22.predict(X22_test)

# Calculate mse
mse22 = mean_squared_error(Y22_test, predicted_y22)

# calculate rmse
rmse22 = math.sqrt(mse22)

# Print model metrics
print("MSE =", round(mse22, 3))
print("RMSE =", round(rmse22, 3))
len(Y)
```

```
MSE = 0.001
RMSE = 0.026
```

Out[ ]: 1320

C.

The 2022 model peformed .008 runs per-at-bat worse than the hall-of-fame data set. This could be explained by a few things. First, there are only 558 data points for the 2022 data and 1,320 for the hall-of-fame data so there is less data to train on for 2022 potentially explaing the loss in RMSE. There are also differences in style played that could give different dynamics for a long term data set. For example, steroid testing began in 2003 having an impact in the amounts of runs scored. This could have an impact on the relationship of runs and hits given that the data includes 10 years of steroid era baseball. One final difference is that the presence of hall-of-fame eligibility in the hall-of-fame data set means that players played for at least 10 years. The 2022 data will have outlier playeres with very few at-bats making it harder to predict for these players.

Question 5

In [ ]:

```
# Calculate mean runs per-at-bat for hof data
mean_runs_hof = np.mean(hitting_rates_df['runs'])

# Calculate mean runs per-at-bat for 2022 players
mean_runs_2022 = np.mean(players_2022['runs'])

# Print mean results
print("Mean runs per-at-bat HOF =", round(mean_runs_hof, 3))
print("Mean runs per-at-bat 2022 =", round(mean_runs_2022, 3))

# Conduct independent t-test
t_stat, p_value = stats.ttest_ind(hitting_rates_df['runs'], players_2022['runs'])

# Print t-test results
print("T-statistic =", t_stat)
print("P-value =", p_value)
```

```
Mean runs per-at-bat HOF = 0.134
Mean runs per-at-bat 2022 = 0.123
T-statistic = 7.050031609009365
P-value = 2.504267264381346e-12
```

With a p-value << .001, we can conclude that there is a statistically significant difference between runs-at-bat for hall-of-fame eligible players when compared to 2022 players. The mean difference is .011 runs per-at-bat. This is sensible considering the length of time played by hall-of-fame eligible players. To be able to play for 10+ years in the MLB, it would stand to reason that you have demonstrated some competence while players from any given season will include rookies and players who will not last long in the league.