hypothesis

November 24, 2019

[108]:
```
import pandas as pd
import plotly.figure_factory as ff
import plotly.graph_objects as go
import plotly.io as pio
import math
from scipy import stats
# renderer for jupyter notebook
from sklearn.metrics import mean_absolute_error
pio.renderers.default='notebook'
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

[109]:
```
pio.templates.default = "plotly_dark"
```

[110]:
```
df_scorecard=pd.read_csv(r'./full/odi_scorecard.csv')
df_info=pd.read_csv(r'./full/odi_info.csv')
```

0.0.1 Hypothesis

1

- H(0):Mean value of batsman bowled is equal to mean value of batsman dismissed by lbw in ODI
- H(A):Mean value of batsman bowled is not equal to mean value batsman dismissed by lbw

Data

[111]:
```
df_first=df_scorecard[(df_scorecard['wicket-method']=='bowled')|(df_scorecard['wicket-method']
```

[112]:
```
df_first['lbw']=df_first['wicket-method'].apply(lambda x: 0 if x=='bowled' else␣
↪1 )
df_first['bowled']=df_first['wicket-method'].apply(lambda x: 1 if x=='bowled'␣
↪else 0 )
```

[113]:
```
df_first=df_first.groupby(['match-id'],as_index=False).sum()
```

[114]:
```
df_first=df_first[['lbw','bowled']]
```

Visualizations

[116]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_first['lbw'], histnorm='probability',␣
 ↪name='lbw'))
fig.add_trace(go.Histogram(x=df_first['bowled'],␣
 ↪histnorm='probability',name='bowled'))
fig.update_layout(title='Probability distribution for lbw and␣
 ↪bowled',xaxis_title='Number of wickets',yaxis_title='Probability')
fig.show()
```

[117]:
```
fig=ff.
 ↪create_distplot([df_first['lbw'],df_first['bowled']],['LBW','Bowled'],bin_size=1,curve_type
fig.update_layout(title_text='Distribution of dimissal␣
 ↪methods',xaxis_title='Number of wickets',yaxis_title='Density')
fig.show()
```

Hypothesis Testing

Paired T test

[118]:
```
df_first[['lbw','bowled']].describe()
```

[118]:

	lbw	bowled
count	1677.000000	1677.000000
mean	1.679785	2.774001
std	1.386744	1.684788
min	0.000000	0.000000
25%	1.000000	2.000000
50%	1.000000	3.000000
75%	2.000000	4.000000
max	8.000000	9.000000

[119]:
```
ttest,pval=stats.ttest_rel(df_first['lbw'],df_first['bowled'])
```

[120]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
8.855848765261422e-83
reject null hypothesis
```

2

- H(0):Wickets fallen in the first 70% of the first innings is equal to the wickets fallen in the last 30% of the first innings
- H(A):Wickets fallen in the first 70% of the first innings is not equal to the wickets fallen in the last 30% of the first innings

- H(0):Wickets fallen in the first 71% of the first innings is equal to the wickets fallen in the last 29% of the first innings
- H(A):Wickets fallen in the first 71% of the first innings is not equal to the wickets fallen in the last 29% of the first innings
- H(0):Wickets fallen in the first 50% of the first innings is equal to the wickets fallen in the last 50% of the first innings
- H(A):Wickets fallen in the first 50% of the first innings is not equal to the wickets fallen in the last 50% of the first innings
- H(0):Wickets fallen in the first 10 overs of the first innings is equal to the wickets fallen in the last 10 overs of the first innings
- H(A):Wickets fallen in the first 10 overs of the first innings is not equal to the wickets fallen in the last 10 overs of the first innings

Data

[121]:
```
temp=pd.DataFrame(data=None)
temp=df_scorecard[df_scorecard['innings']==1]
temp=temp.groupby('match-id',as_index=False).sum()
temp['total-overs']=round(temp['balls-played']/6)
temp=temp[['match-id','total-overs']]
temp['first-seventy']=round(temp['total-overs']*0.70)
temp['first-seventyone']=round(temp['total-overs']*0.71)
temp['first-fifty']=round(temp['total-overs']*0.5)
temp['last-ten']=round(temp['total-overs']-10)
temp=temp.merge(df_scorecard[(df_scorecard['innings']==1)␣
 ↪&(df_scorecard['fall-of-wicket-overs']>0.0)],on=['match-id'])
temp['fall-of-wicket-overs']=temp['fall-of-wicket-overs'].apply(lambda x:␣
 ↪int(x)+1)
```

[122]:
```
df_second=pd.DataFrame({'match-id':temp['match-id']})
df_second['first-seventy-wickets']=temp[(temp['fall-of-wicket-overs']<=temp['first-seventy'])␣
 ↪& (temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
df_second['last-thirty-wickets']=temp[temp['fall-of-wicket-overs']>temp['first-seventy']]['fal
df_second['first-seventy-wickets']=df_second['first-seventy-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)
df_second['last-thirty-wickets']=df_second['last-thirty-wickets'].apply(lambda␣
 ↪x:1 if x>0 else 0)

df_second['first-seventyone-wickets']=temp[(temp['fall-of-wicket-overs']<=temp['first-seventyo
 ↪& (temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
df_second['last-twentynine-wickets']=temp[temp['fall-of-wicket-overs']>temp['first-seventyone'
df_second['first-seventyone-wickets']=df_second['first-seventyone-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)
df_second['last-twentynine-wickets']=df_second['last-twentynine-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)

df_second['first-fifty-wickets']=temp[(temp['fall-of-wicket-overs']<=temp['first-fifty'])␣
 ↪& (temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
```

```
df_second['last-fifty-wickets']=temp[temp['fall-of-wicket-overs']>temp['first-fifty']]['fall-o
df_second['first-fifty-wickets']=df_second['first-fifty-wickets'].apply(lambda␣
 ↪x:1 if x>0 else 0)
df_second['last-fifty-wickets']=df_second['last-fifty-wickets'].apply(lambda x:
 ↪1 if x>0 else 0)

df_second['first-ten-overs-wickets']=temp[(temp['fall-of-wicket-overs']<=10) &␣
 ↪(temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
df_second['last-ten-overs-wickets']=temp[temp['fall-of-wicket-overs']>temp['last-ten']]['fall-
df_second['first-ten-overs-wickets']=df_second['first-ten-overs-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)
df_second['last-ten-overs-wickets']=df_second['last-ten-overs-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)

df_second=df_second.groupby(['match-id'],as_index=False).sum()
df_second=df_second.drop(['match-id'],axis=1)
```

Visualizations

[124]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-seventy-wickets'],␣
 ↪histnorm='probability', name='First 70%'))
fig.add_trace(go.Histogram(x=df_second['last-thirty-wickets'],␣
 ↪histnorm='probability',name='Last 30%'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪70% and last 30% of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[125]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-seventyone-wickets'],␣
 ↪histnorm='probability', name='First 71%'))
fig.add_trace(go.Histogram(x=df_second['last-twentynine-wickets'],␣
 ↪histnorm='probability',name='Last 29%'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪71% and last 29% of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[126]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-fifty-wickets'],␣
 ↪histnorm='probability', name='First 50%'))
fig.add_trace(go.Histogram(x=df_second['last-fifty-wickets'],␣
 ↪histnorm='probability',name='Last 50%'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪50% and last 50% of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
```

```
fig.show()
```

[127]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-ten-overs-wickets'],␣
 ↪histnorm='probability', name='First 10'))
fig.add_trace(go.Histogram(x=df_second['last-ten-overs-wickets'],␣
 ↪histnorm='probability',name='Last 10'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪10 overs and last 10 oversof first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[128]:
```
fig=ff.
 ↪create_distplot([df_second['first-seventy-wickets'],df_second['last-thirty-wickets']],['Fir
 ↪70%','Last 30%'],curve_type='normal')
fig.update_layout(title='Distribution for wickets fallen in first 70% and last␣
 ↪30% of first innings',xaxis_title='Number of wickets',yaxis_title='Density')
fig.show()
```

[129]:
```
fig=ff.
 ↪create_distplot([df_second['first-fifty-wickets'],df_second['last-fifty-wickets']],['First␣
 ↪50%','Last 50%'],curve_type='normal')
fig.update_layout(title='Distribution for wickets fallen in first 50% and last␣
 ↪50% of first innings',xaxis_title='Number of wickets',yaxis_title='Density')
fig.show()
```

[130]:
```
fig=ff.
 ↪create_distplot([df_second['first-ten-overs-wickets'],df_second['last-ten-overs-wickets']],
 ↪10 overs','Last 10 overs'],curve_type='normal')
fig.update_layout(title='Distribution for wickets fallen in first 10 overs and␣
 ↪last 10 overs of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Density')
fig.show()
```

Hypothesis Testing

Paired T Test

[131]:
```
df_second.describe()
```

[131]:

	first-seventy-wickets	last-thirty-wickets	first-seventyone-wickets	\
count	1707.000000	1707.000000	1707.000000	
mean	3.950791	4.100762	4.062097	
std	1.650583	1.652500	1.655468	
min	0.000000	0.000000	0.000000	
25%	3.000000	3.000000	3.000000	
50%	4.000000	4.000000	4.000000	
75%	5.000000	5.000000	5.000000	
max	9.000000	9.000000	9.000000	

```
       last-twentynine-wickets  first-fifty-wickets  last-fifty-wickets  \
count              1707.000000          1707.000000         1707.000000
mean                  3.989455             2.844757            5.206796
std                   1.647374             1.447369            1.790623
min                   0.000000             0.000000            0.000000
25%                   3.000000             2.000000            4.000000
50%                   4.000000             3.000000            5.000000
75%                   5.000000             4.000000            6.000000
max                   9.000000             8.000000           10.000000

       first-ten-overs-wickets  last-ten-overs-wickets
count              1707.000000             1707.000000
mean                  1.325718                3.445226
std                   1.113937                1.594545
min                   0.000000                0.000000
25%                   0.000000                2.000000
50%                   1.000000                3.000000
75%                   2.000000                5.000000
max                   6.000000                9.000000
```

[132]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-seventy-wickets'],df_second['last-thirty-wickets'])
```

[133]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
0.020019531021699132
reject null hypothesis
```

[134]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-seventyone-wickets'],df_second['last-twentynine-wickets'])
```

[135]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
0.25955086942355377
accept null hypothesis
```

[136]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-fifty-wickets'],df_second['last-fifty-wickets'])
```

[137]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
6.436197034363128e-225
reject null hypothesis
```

[138]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-ten-overs-wickets'],df_second['last-ten-overs-wickets'])
```

[139]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
8.92620766849228e-289
reject null hypothesis
```

3

- H(0):There is an equal probability of wicket by the first category of dismissal and second category of dismissal
- H(A):There is an equal probability of wicket by the first category of dismissal and second category of dismissal

Data

[140]:
```
df_third=df_scorecard[df_scorecard['wicket-method']!='0']
```

[141]:
```
first_cat=['run out','hit wicket','obstructing the field','retired␣
 ↪out','stumped']
second_cat=['caught','bowled','lbw','caught and bowled']
```

[142]:
```
df_third['first-category']=df_third['wicket-method'].apply(lambda x: 1 if x in␣
 ↪first_cat else 0 )
df_third['sec-category']=df_third['wicket-method'].apply(lambda x: 1 if x in␣
 ↪second_cat else 0 )
```

[143]:
```
df_third=df_third[['match-id','first-category','sec-category']]
df_third=df_third.groupby(['match-id'],as_index=False).sum()
df_third['wickets']=df_third['first-category']+df_third['sec-category']
```

[144]:
```
# df_third.loc[:,'wic_batsman']=round(df_third['wic_batsman']/
 ↪df_third['wickets'],3)
# df_third.loc[:,'wic_bowler']=round(df_third['wic_bowler']/
 ↪df_third['wickets'],3)
df_third=df_third[['first-category','sec-category']]
```

Visualizations

[146]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_third['first-category'],␣
 ↪histnorm='probability', name='First Category'))
fig.add_trace(go.Histogram(x=df_third['sec-category'],␣
 ↪histnorm='probability',name='Second Category'))
fig.update_layout(title='Probability distribution for wickets fallen by First␣
 ↪category of dismissal methods and second category',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[147]:
```
fig=ff.
 ↪create_distplot([df_third['first-category'],df_third['sec-category']],['First␣
 ↪Category','Second Category'],curve_type='normal')
fig.update_layout(title='Density of wickets fallen by first and second category␣
 ↪of dismissal methods',xaxis_title='Number of wickets',yaxis_title='Density')
fig.show()
```

Hypothesis Testing

Paired T Test

[148]:
```
df_third.describe()
```

[148]:

	first-category	sec-category
count	1707.000000	1707.000000
mean	1.647920	13.127709
std	1.351782	2.980216
min	0.000000	4.000000
25%	1.000000	11.000000
50%	1.000000	13.000000
75%	2.000000	15.000000
max	7.000000	20.000000

[149]:
```
ttest,pval=stats.ttest_rel(df_third['first-category'],df_third['sec-category'])
```

[150]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
0.0
reject null hypothesis
```

Current

0.0.2 ML

Correlation

[151]:
```
df_kohli=df_scorecard[df_scorecard['name']=='V Kohli']
```

[152]:
```
corr_val=df_kohli.drop(['match-id'],axis=1).corr()
corr_list=[]
for i in range(corr_val.shape[0]):
    corr_list.append(corr_val.iloc[:,i])
fig = go.Figure(data=go.Heatmap(
                    z=corr_list,
                    x=corr_val.columns,
                    y=corr_val.columns))
fig.show()
```

[153]:
```
columns = np.full((corr_val.shape[0],), True, dtype=bool)
for i in range(corr_val.shape[0]):
    for j in range(i+1, corr_val.shape[0]):
        if corr_val.iloc[i,j] >= 0.9:
            if columns[j]:
                columns[j] = False
```

[154]:
```
selected_columns = corr_val.columns[columns]
data = df_scorecard[selected_columns]
```

[156]:
```
x = df_kohli.loc[:,'balls-played'].values
y = df_kohli.loc[:, 'runs-scored'].values
```

[157]:
```
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size = 1/3,␣
 ↪random_state = 0)
```

[158]:
```
linearRegressor = LinearRegression()
```

[159]:
```
yTrain = yTrain.reshape(1, -1)
xTrain = xTrain.reshape(1, -1)
yTest = yTest.reshape(1, -1)
xTest = xTest.reshape(1, -1)
```

[160]:
```
linearRegressor.fit(xTrain, yTrain)
```

[160]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[ ]:
```
yPrediction = linearRegressor.predict(xTest)
```

[ ]:
```

```