LogisticRegression

July 14, 2025

[43]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
%matplotlib inline
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
import warnings
warnings.filterwarnings('ignore')

np.random.seed(50)

n = 200 #this will represent the randomized factors for attorney's I'll␣
 ↪randomize below

data = {

'experience': np.random.randint(0, 31, size=n),
'win_ratio' : np.random.uniform(0.2, 0.9, size=n), #ratio strictly between 0.
 ↪2-0.9
'education_level': np.random.choice(['JSD', 'LLM', 'JD'], size=n, p=[0.1, 0.3,␣
 ↪0.6])
}

df = pd.DataFrame(data)
df['education_factored'] = np.array([1.0 if ed == 'JSD' else 0.85 if ed ==␣
 ↪'LLM' else 0.7 for ed in df['education_level']])
df['experience_factored'] = df['experience'] / 30
score = 0.5 * df['experience_factored'] + 0.3 * df['win_ratio'] + 0.2 *␣
 ↪df['education_factored']
df['case_result'] = np.where(score > 0.6, 1, 0)

X = df[['experience','win_ratio','education_factored']]
y = df['case_result']
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)
y_prob = model.predict_proba(x_test)[:,1]
df['probability_predicted_win'] = model.predict_proba(X)[:, 1]
df['probability_predicted_binary_format'] = model.predict(X)

experience_coefficient, win_ratio_coefficient, education_coefficient = model.coef_[0]
intercept = model.intercept_[0]
mean_win_ratio = df['win_ratio'].mean()
mean_education = df['education_factored'].mean()
experience_range = np.linspace(0,30,100)
log = (experience_coefficient * experience_range + win_ratio_coefficient * mean_win_ratio + education_coefficient * mean_education + intercept)
probability = 1 / (1 + np.exp(-log))
threshold = (-(win_ratio_coefficient * mean_win_ratio+education_coefficient*mean_education+intercept))/experience_coefficient

pd.set_option('display.max_rows', None)        # rows
pd.set_option('display.max_columns', None)     # columns
pd.set_option('display.width', None)           #  line width
pd.set_option('display.colheader_justify', 'left')  # align headers
pd.set_option('display.float_format', '{:.6f}'.format)  # format floats
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', None)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
print(df[['experience', 'win_ratio', 'education_level', 'probability_predicted_win', 'probability_predicted_binary_format']])
print("threshold", threshold)
#histogram to show number of lawyers and their probability of winning
plt.figure(figsize=(8, 5))
plt.hist(df['probability_predicted_win'], bins=20, edgecolor='black')
plt.title('Analysis of Predicted Win Probability')
plt.xlabel('Win Probability')
plt.ylabel('Number of Lawyers')
```

```python
plt.grid(True)
plt.show()
#scatter plot to relate experience to win probability
plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='experience', y='probability_predicted_win',␣
 ↪hue='case_result', palette='muted')
plt.title('Experience vs Predicted Win Probability')
plt.xlabel('Experience Quantity')
plt.ylabel('Predicted Probability of Winning')
plt.legend(title='Actual Case Result')
plt.grid(True)
plt.show()
# sigmoid curve
plt.figure(figsize=(8, 5))
plt.plot(experience_range, probability, color='green', label='Sigmoid Curve')
plt.axhline(0.5, linestyle='--', color='black', label='Threshold: 0.5')
plt.axvline(threshold, linestyle = '--', color = 'grey', label = f'Experience␣
 ↪Based Threshold = {threshold:.2f}')
plt.title('Sigmoid Curve: Win Probability with Focus on Experience Level')
plt.xlabel('Experience')
plt.ylabel('Win Probability')
plt.grid(True)
plt.legend()
plt.show()
```

```
Accuracy: 0.95
Confusion Matrix:
 [[36  2]
 [ 1 21]]
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.95      0.96        38
           1       0.91      0.95      0.93        22

    accuracy                           0.95        60
   macro avg       0.94      0.95      0.95        60
weighted avg       0.95      0.95      0.95        60

Coefficients: [[0.40642916 2.83949469 0.54648283]]
Intercept: [-9.23735947]
     experience  win_ratio education_level  probability_predicted_win
probability_predicted_binary_format
0    16          0.724892     JD           0.427116                    0
1     0          0.488699     JD           0.000571                    0
2    11          0.593151     JD           0.062982                    0
3    13          0.367916    LLM           0.079836                    0
```

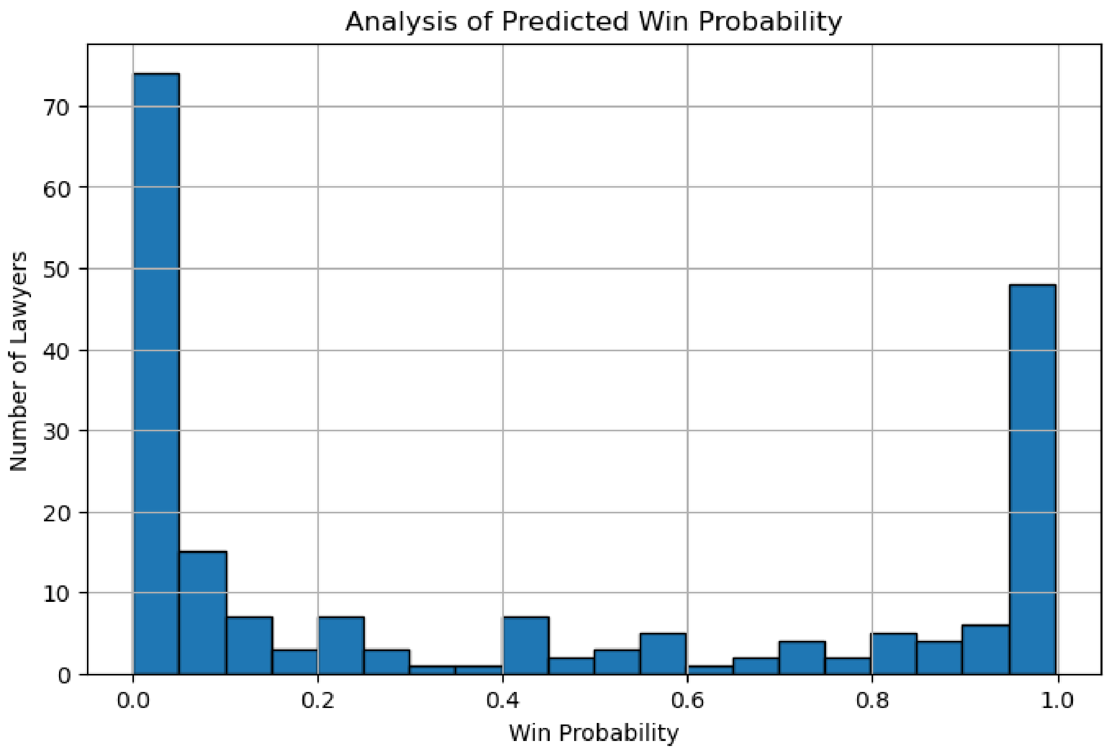
4	1	0.383250	JD	0.000636	0
5	30	0.560522	LLM	0.993384	1
6	4	0.239319	JD	0.001429	0
7	6	0.652204	JD	0.010310	0
8	5	0.734177	LLM	0.009415	0
9	6	0.442228	JD	0.005706	0
10	22	0.745961	LLM	0.907773	1
11	13	0.346041	JD	0.069872	0
12	5	0.220540	JD	0.002032	0
13	2	0.533106	LLM	0.001584	0
14	26	0.630149	JD	0.970734	1
15	7	0.595765	LLM	0.014257	0
16	15	0.255297	JSD	0.133597	0
17	4	0.865157	JD	0.008388	0
18	14	0.699035	JD	0.235069	0
19	3	0.357043	LLM	0.001443	0
20	28	0.814011	JD	0.992128	1
21	27	0.315825	JD	0.953272	1
22	26	0.447229	JD	0.951765	1
23	26	0.891503	JD	0.985850	1
24	6	0.275268	JD	0.003559	0
25	20	0.391752	JD	0.595353	1
26	11	0.554231	JD	0.056767	0
27	17	0.558263	JD	0.410877	0
28	21	0.734373	JD	0.853890	1
29	10	0.794811	JSD	0.085511	0
30	9	0.597730	LLM	0.031746	0
31	0	0.614917	JD	0.000817	0
32	30	0.619788	JD	0.993928	1
33	27	0.281510	JD	0.948735	1
34	6	0.862519	JD	0.018576	0
35	19	0.438782	JD	0.528283	1
36	2	0.298621	LLM	0.000815	0
37	15	0.295244	JD	0.127857	0
38	30	0.804782	JD	0.996400	1
39	9	0.228966	LLM	0.011376	0
40	3	0.885978	JSD	0.006993	0
41	19	0.318494	LLM	0.463482	0
42	26	0.730577	JD	0.977835	1
43	28	0.599077	JSD	0.987754	1
44	19	0.362822	JD	0.474413	0
45	2	0.843999	JD	0.003521	0
46	12	0.674727	JD	0.112867	0
47	0	0.886450	JD	0.001765	0
48	3	0.796397	JD	0.004613	0
49	2	0.566336	JD	0.001604	0
50	10	0.677156	LLM	0.058098	0
51	16	0.337545	LLM	0.212232	0

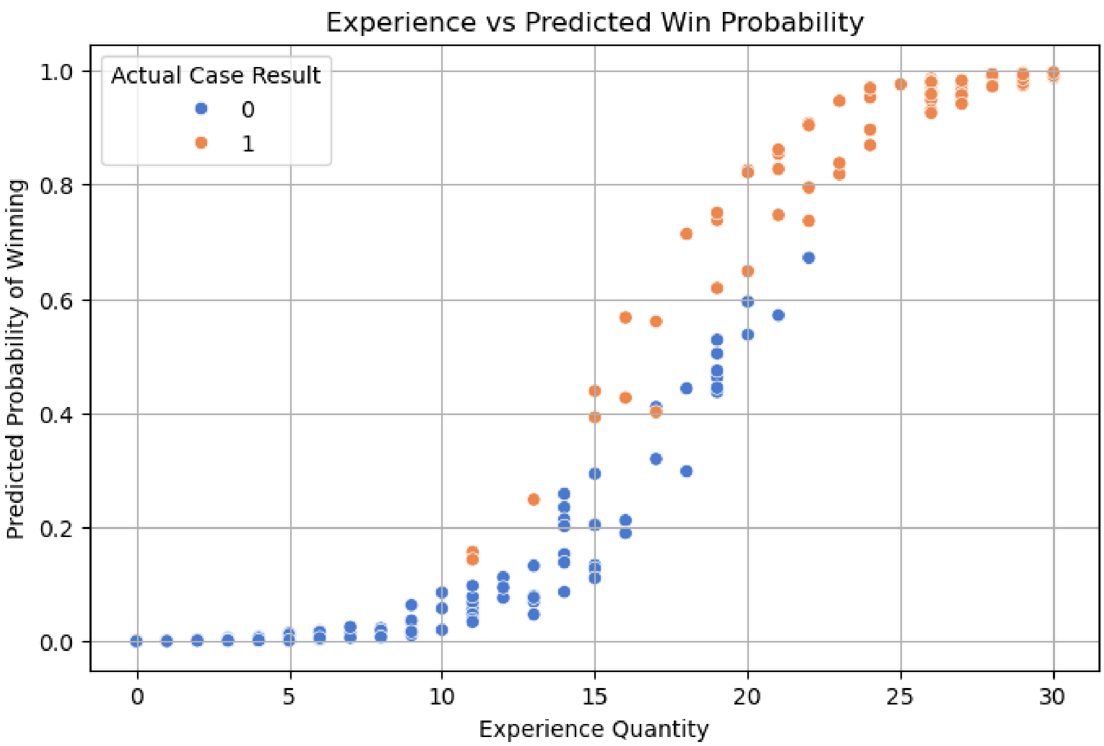
52	14	0.512060	JD	0.153057	0
53	19	0.280936	LLM	0.437090	0
54	0	0.218444	JD	0.000265	0
55	29	0.805449	JD	0.994615	1
56	29	0.219099	LLM	0.974312	1
57	30	0.434040	JD	0.989753	1
58	0	0.421149	LLM	0.000512	0
59	11	0.633866	JD	0.070159	0
60	30	0.663573	LLM	0.995054	1
61	26	0.424825	JD	0.948759	1
62	7	0.617455	JD	0.013973	0
63	19	0.540690	LLM	0.618831	1
64	28	0.872415	JD	0.993323	1
65	24	0.350514	JD	0.869298	1
66	27	0.537864	JD	0.974569	1
67	14	0.656255	JD	0.213933	0
68	30	0.400065	JD	0.988727	1
69	13	0.356106	LLM	0.077407	0
70	7	0.369626	JD	0.006962	0
71	4	0.592831	JD	0.003889	0
72	4	0.494437	LLM	0.003194	0
73	0	0.527928	LLM	0.000693	0
74	26	0.770595	LLM	0.981700	1
75	11	0.430375	JSD	0.047511	0
76	0	0.577841	LLM	0.000798	0
77	13	0.839227	LLM	0.248563	0
78	19	0.404788	JD	0.504182	1
79	26	0.746986	JD	0.978822	1
80	13	0.201654	JD	0.047487	0
81	19	0.735486	LLM	0.738409	1
82	1	0.332477	JSD	0.000648	0
83	29	0.293946	JD	0.977385	1
84	9	0.672280	JD	0.035984	0
85	29	0.508646	JD	0.987580	1
86	20	0.803563	JD	0.825703	1
87	11	0.415884	JD	0.039045	0
88	5	0.518597	JSD	0.005562	0
89	23	0.839880	JD	0.946742	1
90	0	0.743502	JD	0.001177	0
91	19	0.787723	JD	0.751022	1
92	15	0.885077	JD	0.439002	0
93	29	0.427083	LLM	0.985604	1
94	5	0.677183	LLM	0.008019	0
95	28	0.324146	JSD	0.973649	1
96	6	0.441296	JD	0.005691	0
97	1	0.554374	LLM	0.001121	0
98	4	0.419708	LLM	0.002585	0
99	20	0.391510	JD	0.595187	1

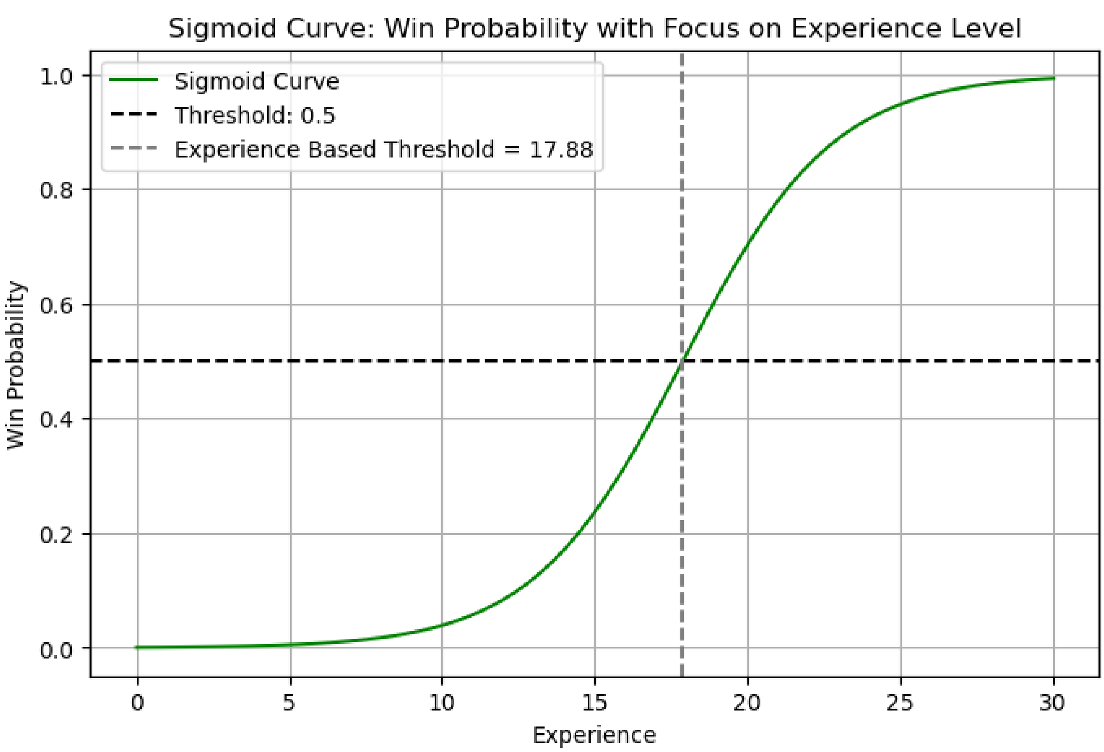
100	4	0.311218	LLM	0.001901	0
101	12	0.524299	JD	0.076638	0
102	21	0.664723	JD	0.827451	1
103	14	0.287192	JD	0.087119	0
104	4	0.451355	LLM	0.002828	0
105	22	0.730547	LLM	0.904043	1
106	11	0.679523	JD	0.079103	0
107	19	0.320603	JD	0.444649	0
108	0	0.452755	JD	0.000516	0
109	5	0.203667	JD	0.001938	0
110	8	0.442307	LLM	0.013851	0
111	30	0.587644	JSD	0.994351	1
112	30	0.741229	JD	0.995691	1
113	12	0.606614	JD	0.094902	0
114	11	0.369993	JD	0.034439	0
115	3	0.533841	JD	0.002194	0
116	6	0.850141	JD	0.017946	0
117	27	0.420139	JD	0.964830	1
118	18	0.240777	JD	0.298299	0
119	26	0.558225	JD	0.964340	1
120	24	0.443242	JD	0.896420	1
121	28	0.363946	JD	0.972310	1
122	8	0.657570	JD	0.023289	0
123	15	0.494909	JD	0.205365	0
124	5	0.871824	JSD	0.015021	0
125	26	0.762892	JD	0.979739	1
126	20	0.442844	LLM	0.648667	1
127	23	0.356517	JD	0.818375	1
128	29	0.891752	LLM	0.996111	1
129	24	0.744235	JD	0.953146	1
130	15	0.210091	LLM	0.111070	0
131	4	0.715611	JD	0.005502	0
132	4	0.666102	LLM	0.005190	0
133	18	0.433189	LLM	0.443471	0
134	15	0.662574	JD	0.293798	0
135	10	0.825921	LLM	0.086010	0
136	17	0.390652	LLM	0.319885	0
137	26	0.312850	JD	0.930905	1
138	8	0.593880	JD	0.019511	0
139	14	0.413221	JSD	0.138533	0
140	11	0.761429	JD	0.097790	0
141	27	0.340543	JD	0.956301	1
142	27	0.351091	JD	0.957535	1
143	27	0.672945	JD	0.982529	1
144	7	0.807582	LLM	0.025713	0
145	9	0.681104	JD	0.036864	0
146	1	0.621343	JD	0.001249	0
147	21	0.494124	JD	0.747109	1

148	8	0.236191	LLM	0.007762	0
149	3	0.256882	JSD	0.001179	0
150	3	0.390863	JSD	0.001723	0
151	21	0.213861	JD	0.571371	1
152	10	0.324009	JD	0.020422	0
153	3	0.213260	JD	0.000884	0
154	6	0.794477	LLM	0.016653	0
155	24	0.843274	JSD	0.969466	1
156	15	0.462948	LLM	0.203936	0
157	22	0.274030	JSD	0.736658	1
158	0	0.546048	LLM	0.000730	0
159	24	0.871341	LLM	0.969398	1
160	18	0.835240	LLM	0.713927	1
161	26	0.587159	JD	0.967060	1
162	11	0.893677	JSD	0.156756	0
163	30	0.359352	JD	0.987363	1
164	1	0.553599	JD	0.001031	0
165	17	0.741983	LLM	0.560528	1
166	29	0.713852	LLM	0.993572	1
167	14	0.743748	JD	0.258660	0
168	26	0.532923	JD	0.961785	1
169	22	0.447092	JD	0.795131	1
170	14	0.601899	LLM	0.202015	0
171	5	0.792625	JSD	0.012032	0
172	21	0.756655	JD	0.861608	1
173	6	0.449504	LLM	0.006319	0
174	20	0.308557	JD	0.537407	1
175	30	0.441370	LLM	0.990745	1
176	30	0.468149	JD	0.990690	1
177	9	0.885478	JD	0.064005	0
178	4	0.769396	JD	0.006404	0
179	25	0.842925	JD	0.975859	1
180	30	0.831813	LLM	0.996927	1
181	9	0.410677	JD	0.017449	0
182	27	0.205177	LLM	0.941769	1
183	0	0.512889	JD	0.000612	0
184	26	0.740825	LLM	0.980118	1
185	15	0.818170	JD	0.392886	0
186	16	0.317462	JD	0.189921	0
187	26	0.284939	JD	0.925630	1
188	23	0.376352	LLM	0.838034	1
189	17	0.516434	LLM	0.401999	0
190	16	0.895134	LLM	0.567522	1
191	20	0.793135	JD	0.821401	1
192	26	0.446479	JSD	0.958674	1
193	4	0.393421	JD	0.002211	0
194	11	0.885906	LLM	0.143490	0
195	1	0.462958	JD	0.000797	0

```
196   5          0.211281    LLM            0.002149                  0
197  13          0.539063    JSD            0.132776                  0
198  22          0.222397     JD            0.672192                  1
199   3          0.493272     JD            0.001956                  0
threshold 17.87769296753244
```




Experience vs Predicted Win Probability
Actual Case Result
0
1
Predicted Probability of Winning
Experience Quantity
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30


Sigmoid Curve: Win Probability with Focus on Experience Level
Sigmoid Curve
Threshold: 0.5
Experience Based Threshold = 17.88
Win Probability
Experience
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30

```
[ ]:
```

```
[ ]:
```