hw4

October 12, 2024

1 Question 3

1.1 (a)

K-fold cross validation is implemented by:

1) Dividing dataset into k groups of len n/k.
2) Save the first fold as the validation set.
3) Compute the MSE for all other folds, leaving out df[k], the validation set
4) Calculate the mean of all MSEs to get an estimate of the k-fold CV.

1.2 (b)

i. Validation Set

K-fold CV has less variance as it's estimate is generated over k folds. Validation sets are a simple and lightweight computation. It is only trained once but it leaves room for high variance as a training set is not being utilized. This affects small datasets. The bias is also strong if the validation set does not properly represent the overall dataset.

ii. LOOCV

K-fold CV has better computational performance than LOOCV as n increases. The bias-variance trade off also helps propogate off variance and introducing bias as the validation set is left out. LOOCV works best on small datasets as it is more flexible and can be more precise.

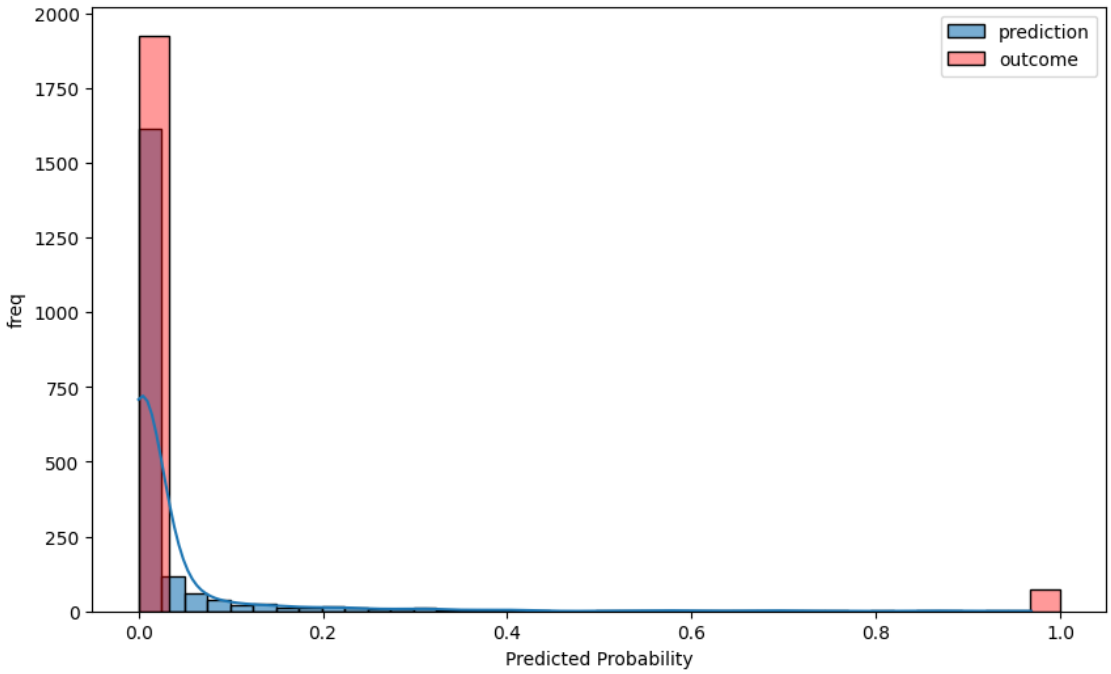
2 Question 6

2.1 (a)

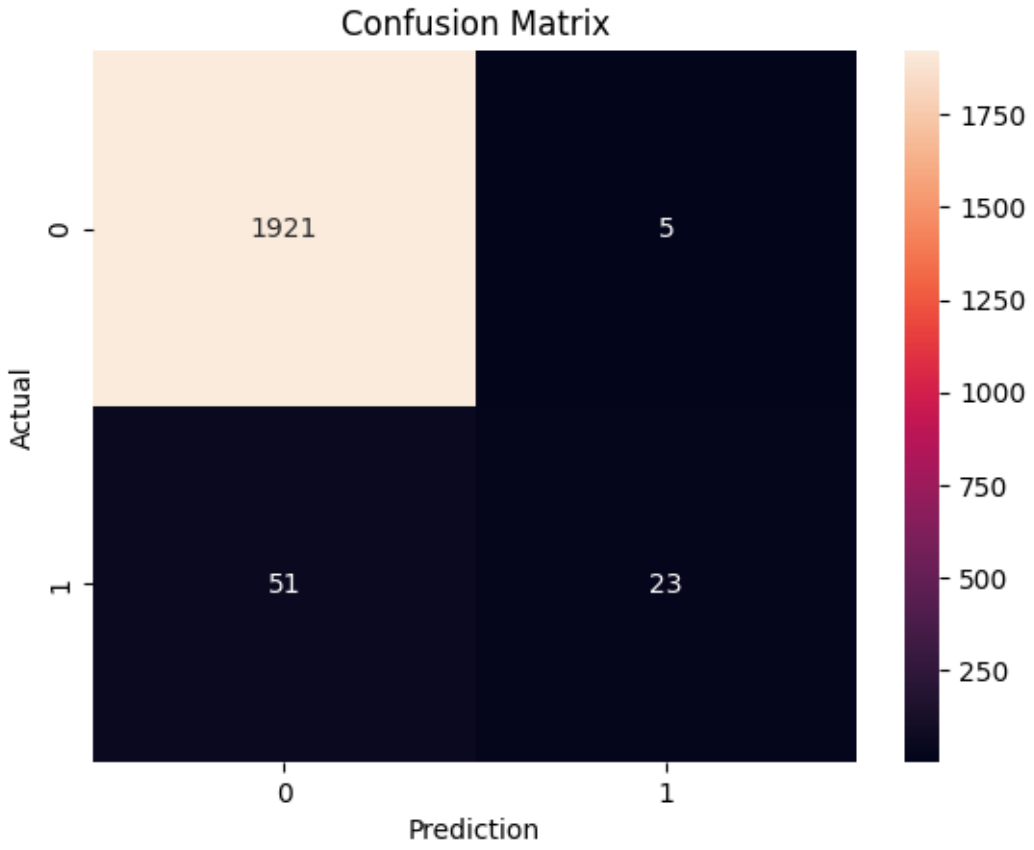
[1]:
```
import pandas as pd
import statsmodels.api as sm
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

default = pd.read_csv('Default.csv')
```

```
default['default'] = default['default'].map({'Yes': 1, 'No': 0})

X = default[['income', 'balance']]
y = default['default']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

model = sm.GLM(y_train, X_train, family=sm.families.Binomial()).fit()

y_pred_prob = model.predict(X_test)

# Converting to y/n
y_pred = (y_pred_prob >= 0.5).astype(int)
```

[2]:
```
plt.figure(figsize=(10,6))
sns.histplot(y_pred_prob, bins=39, kde=True, alpha=0.6, label='prediction')
sns.histplot(y_test, bins=30, kde=False, color='red', alpha=0.4,␣
 ↪label='outcome')
plt.xlabel('Predicted Probability')
plt.ylabel('freq')
plt.legend()
plt.show()
```



[3]:
```
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Prediction')
plt.ylabel('Actual')
plt.show()
```



$Pr(0) accuracy$: 97.4%

$Pr(1) accuracy$: 82.1%

$Pr(0) inaccuracy$: 2.6%

$Pr(1) inaccuracy$: 17.9%

$Overall Accuracy$: 97.2%

[4]:
```
model.summary()
```

[4]:

Dep. Variable:	default	**No. Observations:**	8000
Model:	GLM	**Df Residuals:**	7997
Model Family:	Binomial	**Df Model:**	2
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-623.83
Date:	Fri, 11 Oct 2024	**Deviance:**	1247.7
Time:	19:04:55	**Pearson chi2:**	3.80e+03
No. Iterations:	9	**Pseudo R-squ. (CS):**	0.1218
Covariance Type:	nonrobust		

	coef	std err	z	P> |z|	[0.025	0.975]
const	-11.2316	0.477	-23.543	0.000	-12.167	-10.297
income	1.558e-05	5.6e-06	2.783	0.005	4.61e-06	2.66e-05
balance	0.0055	0.000	22.133	0.000	0.005	0.006

$Y = -11.23 + 1.56 * 10^{-5} income + 0.0056 balance$

2.2 (b)

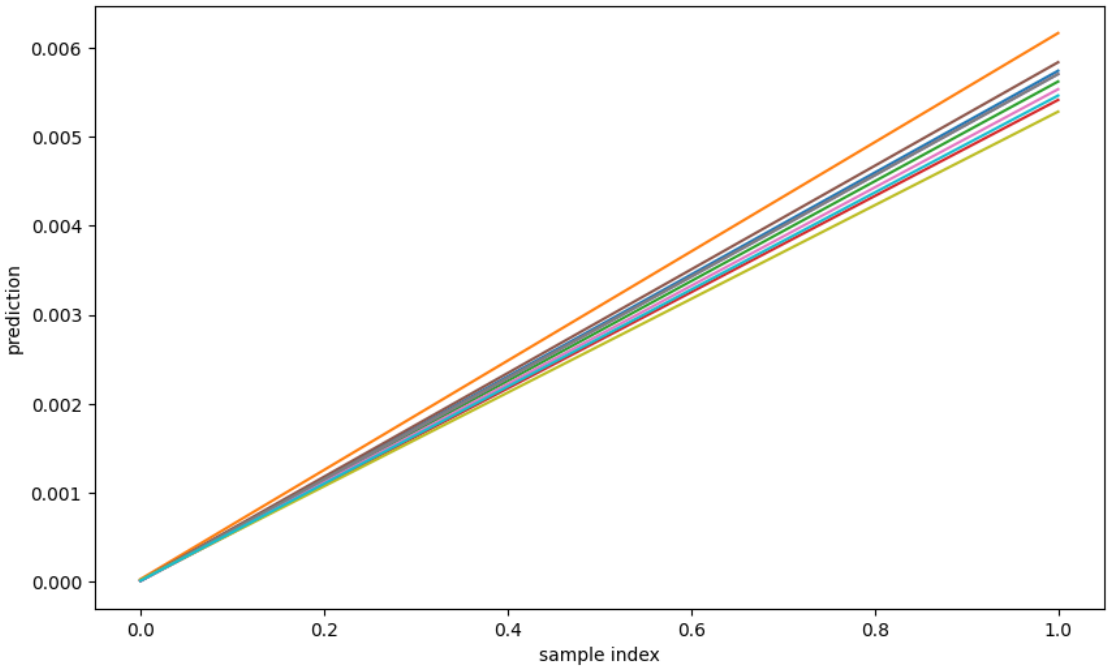
[5]:
```
'''
    Fits a logistic regression model to the Default dataset
    using the specified index and returns the coefficient estimates
    for income and balance.

    Parameters:
    - data:<DataFrame>
    - index:<List> Of indices for resampling the dataset

    Return:
        [coef.income, coef.balance]
'''
def boot_fn(data, index):

    resampled_data = data.iloc[index]

    X = resampled_data[['income', 'balance']]
    y = resampled_data['default']

    X = sm.add_constant(X)

    model = sm.GLM(y, X, family=sm.families.Binomial()).fit()

    return model.params[['income', 'balance']]
```

2.3 (c)

[6]:
```
from functools import partial
np.random.seed(1)
index = np.random.choice(a=len(default), size=len(default), replace=True)

rng = np.random.default_rng(0)
partial = partial(boot_fn, default)

res = np.array([partial(rng.choice(len(default), len(default), replace=True))
                for _ in range(10)])
stderr = np.std(res, axis=0)
print(f'STD Income: {stderr[0]} \nSTD Balance: {stderr[1]}')
# boot_fn(default, index)
```

```
STD Income: 5.555094577549571e-06
STD Balance: 0.00023526954529095226
```

[7]:
```
plt.figure(figsize=(10,6))
for i, res in enumerate(res):
    plt.plot(res, label=f'Sample {i+1}')

plt.xlabel('sample index')
plt.ylabel('prediction')
plt.show()
```



3 (d)

Standard Errors

Bootstrapping: - Income: 5.56e-06 - Balance: 2.35e-04

Logistic Regression Model: - Income: 5.6e-06 - Balance: 0

The standard error of income is the same for both methods. For balance, the regression model has an error of zero. This means one of two things: 1. The predictor can 100% accurately predict the output without error 2. The predictor is redundant and can be represented within another predictor

4 Question 7

4.1 (a)

[8]:
```
Weekly = pd.read_csv('Weekly.csv')
Weekly.head()
```

[8]:

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
0	1990	0.816	1.572	-3.936	-0.229	-3.484	0.154976	-0.270	Down
1	1990	-0.270	0.816	1.572	-3.936	-0.229	0.148574	-2.576	Down
2	1990	-2.576	-0.270	0.816	1.572	-3.936	0.159837	3.514	Up
3	1990	3.514	-2.576	-0.270	0.816	1.572	0.161630	0.712	Up
4	1990	0.712	3.514	-2.576	-0.270	0.816	0.153728	1.178	Up

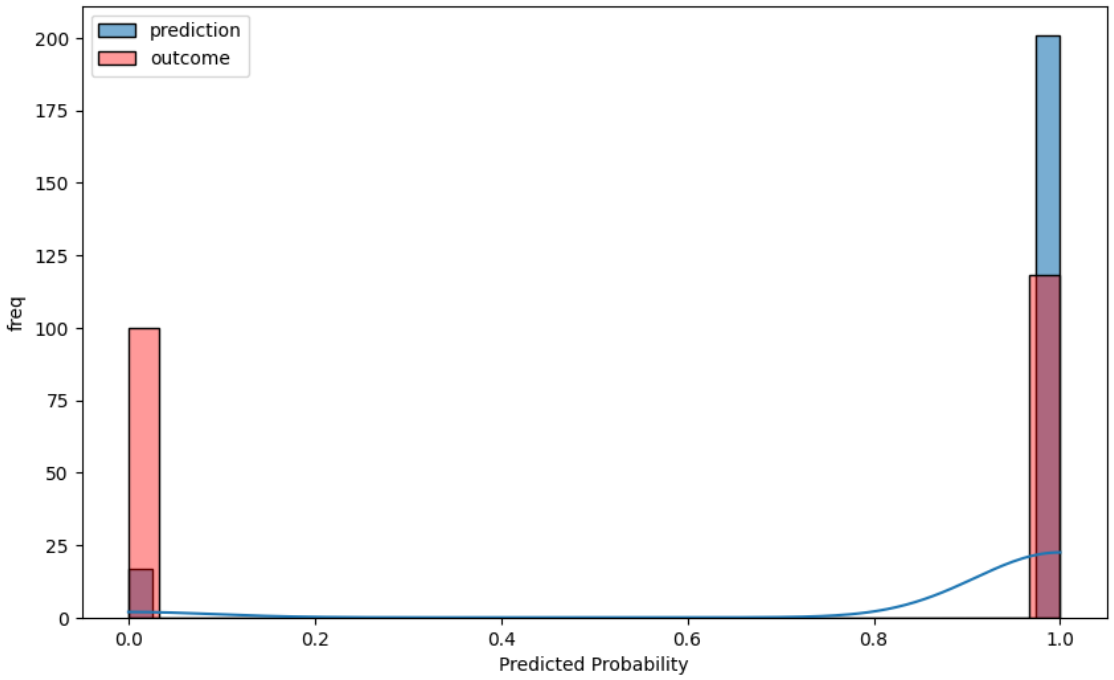
[9]:
```
X = Weekly[['Lag1', 'Lag2']]
y = Weekly['Direction'].map({ 'Up': 1, 'Down': 0})

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

model = sm.GLM(y_train, X_train, family=sm.families.Binomial()).fit()

y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob >= 0.5).astype(int)

model.summary()
```

[9]:

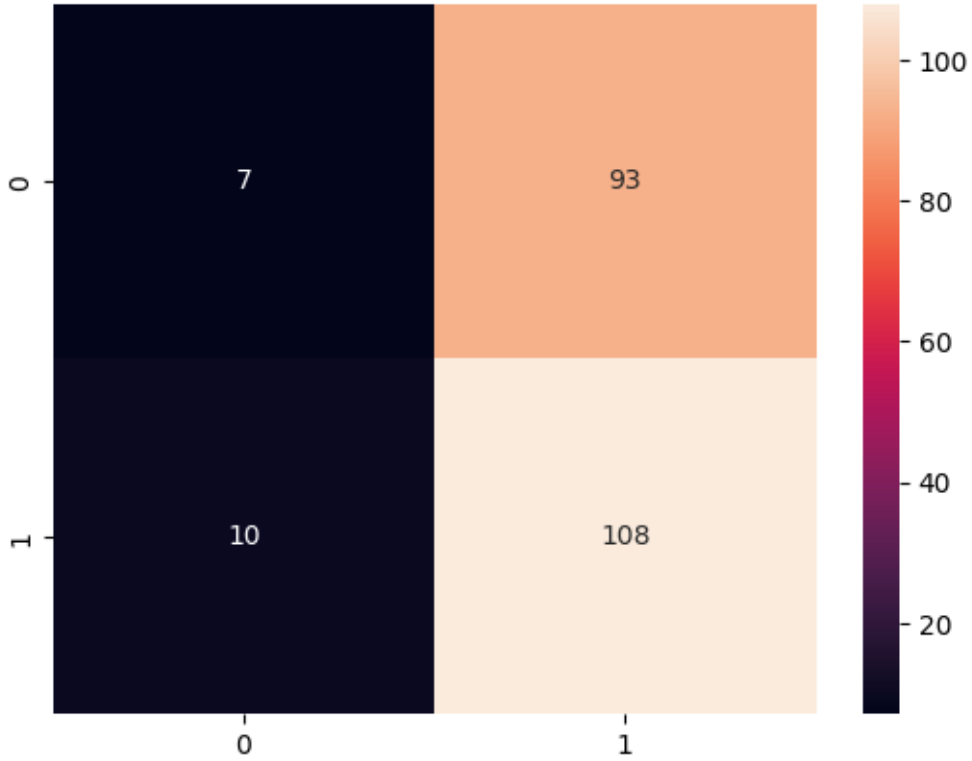
Dep. Variable:	Direction	**No. Observations:**	871
Model:	GLM	**Df Residuals:**	868
Model Family:	Binomial	**Df Model:**	2
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-594.20
Date:	Fri, 11 Oct 2024	**Deviance:**	1188.4
Time:	19:04:55	**Pearson chi2:**	871.
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.007836
Covariance Type:	nonrobust		

	coef	std err	z	P> |z|	[0.025	0.975]
const	0.2295	0.069	3.332	0.001	0.095	0.364
Lag1	-0.0258	0.028	-0.911	0.362	-0.081	0.030
Lag2	0.0727	0.031	2.374	0.018	0.013	0.133

[10]:
```
plt.figure(figsize=(10,6))
sns.histplot(y_pred, bins=39, kde=True, alpha=0.6, label='prediction')
sns.histplot(y_test, bins=30, kde=False, color='red', alpha=0.4,␣
 ↪label='outcome')
plt.xlabel('Predicted Probability')
plt.ylabel('freq')
plt.legend()
plt.show()
```



[11]:
```
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, annot=True, fmt='d')
plt.show()
```



4.2 (b)

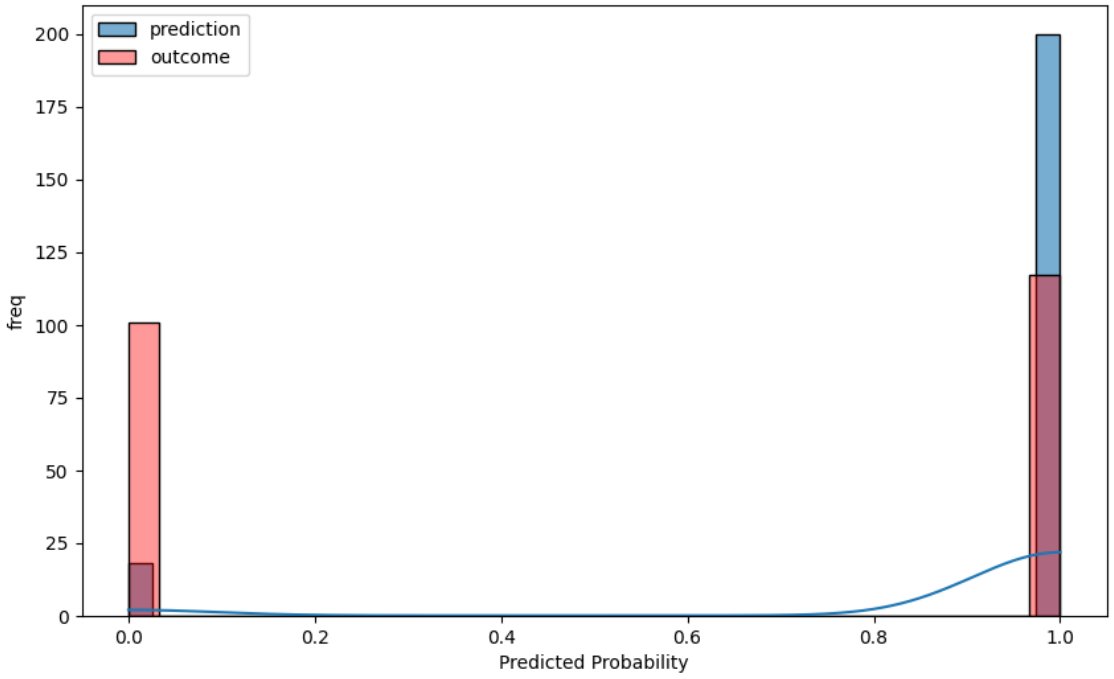
[12]:
```
Weekly1 = Weekly.iloc[1:]
X = Weekly1[['Lag1', 'Lag2']]
y = Weekly1['Direction'].map({'Up': 1, 'Down': 0})

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

model = sm.GLM(y_train, X_train, family=sm.families.Binomial()).fit()

y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob >= 0.5).astype(int)

model.summary()
```

[12]:

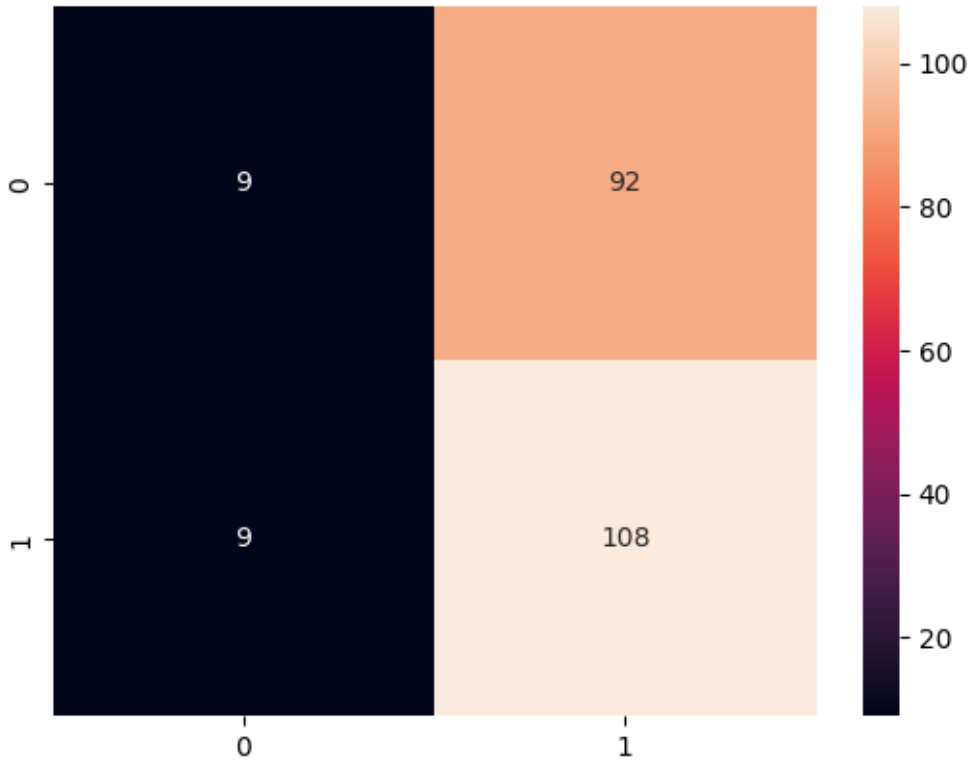
Dep. Variable:	Direction	**No. Observations:**	870
Model:	GLM	**Df Residuals:**	867
Model Family:	Binomial	**Df Model:**	2
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-593.62
Date:	Fri, 11 Oct 2024	**Deviance:**	1187.2
Time:	19:04:55	**Pearson chi2:**	870.
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.006748
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.2451	0.069	3.559	0.000	0.110	0.380
Lag1	-0.0526	0.031	-1.692	0.091	-0.114	0.008
Lag2	0.0517	0.030	1.705	0.088	-0.008	0.111

[13]:
```
plt.figure(figsize=(10,6))
sns.histplot(y_pred, bins=39, kde=True, alpha=0.6, label='prediction')
sns.histplot(y_test, bins=30, kde=False, color='red', alpha=0.4,␣
 ↪label='outcome')
plt.xlabel('Predicted Probability')
plt.ylabel('freq')
plt.legend()
plt.show()
```



[14]:
```
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, annot=True, fmt='d')
plt.show()
```



$Pr(0)accuracy$: 50%

$Pr(1)accuracy$: 54%

$Pr(0)inaccuracy$: 50%

$Pr(1)inaccuracy$: 46%

$OverallAccuracy$: 53.67%

4.3 (c)

[15]:
```
first_observation = Weekly[['Lag1', 'Lag2']].iloc[0:1]

first_observation['const'] = 1.0
first_observation = first_observation[['const', 'Lag1', 'Lag2']]

prediction = model.predict(first_observation).iloc[0]

if prediction > .5:
    print('Up')
else:
    print('Down')
```

Up

[16]:
```
actual_val = Weekly1['Direction'].iloc[0]
predicted_val = 'Up'
accuracy = (predicted_val == ('Up' if actual_val == 1 else 'Down'))
accuracy
```

[16]: False

4.4 The prediction was inaccurate!

4.5 (d)

[40]:
```
err = np.zeros(len(Weekly)) #1089

for i in range(1, len(Weekly)):
    train_data = Weekly.drop(index=i)
    test_data = Weekly.iloc[i : i + 1]

    X_train = train_data[['Lag1', 'Lag2']]
    y_train = train_data['Direction'].map({'Up': 1, 'Down': 0})

    X_test = test_data[['Lag1', 'Lag2']]
    y_test = test_data['Direction']

    model = sm.GLM(y_train, X_train, family=sm.families.Binomial()).fit()

    pred_prob = model.predict(X_test)
    prediction = (pred_prob >= 0.5).astype(int)
    actual = 1 if y_test.values[0] == 'Up' else 0

    # print(f'P:{prediction.item()} \t A:{actual}')
    if (prediction.item() != actual): err[i] = 1
np.sum(err).item()
```

[40]: 505.0

[41]:
```
np.mean(err).item()
```

[41]: 0.4637281910009183

4.5.1 The LOOCV estimate for the test error is 46.37% which can mean the following:

- The model does not fit for our data
- Lag1 and Lag2 are weak predictors