HW2

April 25, 2025

1 - Problem 1

Given:

- Mean profit for companies that issued dividends: $\mu_{\text{Yes}} = 10$
- Mean profit for companies that didn't issue dividends: $\mu_{\text{No}} = 0$
- Variance: $\sigma^2 = 36 \Rightarrow \sigma = 6$
- Prior probabilities:
  - $P(\text{Yes}) = 0.8$
  - $P(\text{No}) = 0.2$

Apply Bayes' theorem:

$$P(\text{Yes} \mid X = 4) = \frac{P(X = 4 \mid \text{Yes}) \cdot P(\text{Yes})}{P(X = 4)}$$

Where:

$$P(X = 4) = P(X = 4 \mid \text{Yes}) \cdot P(\text{Yes}) + P(X = 4 \mid \text{No}) \cdot P(\text{No})$$

Model X based on conditional distributions:

$$P(X = 4 \mid \text{Yes}) = \frac{1}{\sqrt{2\pi \cdot 36}} \exp\left(-\frac{(4-10)^2}{2 \cdot 36}\right) \approx 0.04032845$$

$$P(X = 4 \mid \text{No}) = \frac{1}{\sqrt{72\pi}} \exp\left(-\frac{(4-0)^2}{72}\right) \approx 0.05324133$$

Plug it in:

$$P(\text{Yes} \mid X = 4) = \frac{P(X = 4 \mid \text{Yes}) \cdot P(\text{Yes})}{P(X = 4)}$$

$$P(\text{Yes} \mid X = 4) = \frac{0.04032845 \cdot 0.8}{0.04032845 \cdot 0.8 + 0.05324133 \cdot 0.2}$$

Final Answer:

$$P(\text{Yes} \mid X = 4) \approx 0.752$$

2 - Problem 2

My model:

$$\hat{P}(Y = \text{orange} \mid X = x) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)}$$

Friend's model:

$$\hat{P}(Y = \text{orange} \mid X = x) = \frac{\exp(\hat{\alpha}_0^{\text{orange}} + \hat{\alpha}_1^{\text{orange}} x)}{\exp(\hat{\alpha}_0^{\text{orange}} + \hat{\alpha}_1^{\text{orange}} x) + \exp(\hat{\alpha}_0^{\text{apple}} + \hat{\alpha}_1^{\text{apple}} x)}$$

2.0.1 (a)

$$\log\left(\frac{\hat{P}(\text{orange} \mid x)}{\hat{P}(\text{apple} \mid x)}\right) = \hat{\beta}_0 + \hat{\beta}_1 x$$

2.0.2 (b)

Using the softmax formulation:

$$\log\left(\frac{\hat{P}(\text{orange} \mid x)}{\hat{P}(\text{apple} \mid x)}\right) = (\hat{\alpha}_0^{\text{orange}} + \hat{\alpha}_1^{\text{orange}} x) - (\hat{\alpha}_0^{\text{apple}} + \hat{\alpha}_1^{\text{apple}} x)$$

So the log odds is:

$$(\hat{\alpha}_0^{\text{orange}} - \hat{\alpha}_0^{\text{apple}}) + (\hat{\alpha}_1^{\text{orange}} - \hat{\alpha}_1^{\text{apple}})x$$

2.0.3 (c)

From (a) and (b):

$$\hat{\beta}_0 + \hat{\beta}_1 x = (\hat{\alpha}_0^{\text{orange}} - \hat{\alpha}_0^{\text{apple}}) + (\hat{\alpha}_1^{\text{orange}} - \hat{\alpha}_1^{\text{apple}})x$$

We can pick any constants satisfying:

- $\hat{\alpha}_0^{\text{orange}} - \hat{\alpha}_0^{\text{apple}} = 2$
- $\hat{\alpha}_1^{\text{orange}} - \hat{\alpha}_1^{\text{apple}} = -1$

One possible solution:

- $\hat{\alpha}_0^{\text{orange}} = 2$, $\hat{\alpha}_1^{\text{orange}} = -1$
- $\hat{\alpha}_0^{\text{apple}} = 0$, $\hat{\alpha}_1^{\text{apple}} = 0$

2.0.4 (d)

My friend gets:

- $\hat{\alpha}_0^{\text{orange}} = 1.2$, $\hat{\alpha}_1^{\text{orange}} = -2$
- $\hat{\alpha}_0^{\text{apple}} = 3$, $\hat{\alpha}_1^{\text{apple}} = 0.6$

Then:

$$\hat{\beta}_0 = \hat{\alpha}_0^{\text{orange}} - \hat{\alpha}_0^{\text{apple}} = 1.2 - 3 = -1.8$$

$$\hat{\beta}_1 = \hat{\alpha}_1^{\text{orange}} - \hat{\alpha}_1^{\text{apple}} = -2 - 0.6 = -2.6$$

2.0.5 (e)

Since both models produce the same log odds (i.e., same decision boundary), their predicted class labels will always be the same. Only the probability outputs might differ due to different parametrizations. Therefore, there is 100% agreement between predicted class labels.

3 - Problem 3

Given:

- X_1 = hours studied
- X_2 = undergrad GPA

Then:

$$\hat{P}(Y = 1 \mid X_1, X_2) = \frac{1}{1 + \exp(-(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2))}$$

Given coefficients:

- $\hat{\beta}_0 = -6$
- $\hat{\beta}_1 = 0.05$
- $\hat{\beta}_2 = 1$

3.0.1 (a)

Plug in:

- $X_1 = 40$, $X_2 = 3.7$

Compute linear combination:

$$z = -6 + 0.05 \cdot 40 + 1 \cdot 3.7 = -6 + 2 + 3.7 = -0.3$$

Apply to logistic function:

$$\hat{P}(Y = 1) = \frac{1}{1 + \exp(-(-0.3))} = \frac{1}{1 + \exp(0.3)} \approx \frac{1}{1 + 1.3499} \approx \frac{1}{2.3499} \approx 0.4256$$

Answer:

$$P(\{\text{Get A}\}) \approx 0.426$$

3.0.2 (b)

Given:

- $\hat{P}(Y = 1) = 0.8$
- $X_2 = 3.7$
- $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$

Inverse of the logistic function:

$$0.8 = \frac{1}{1 + \exp(-(-6 + 0.05X_1 + 3.7))}$$

Solve for exponent:

$$\frac{1}{0.8} - 1 = \exp(-(-6 + 0.05X_1 + 3.7)) \Rightarrow 0.25 = \exp(2.3 - 0.05X_1)$$
$$\ln(0.25) = 2.3 - 0.05X_1 \Rightarrow -1.3863 = 2.3 - 0.05X_1$$

Solve for X_1:

$$-1.3863 - 2.3 = -0.05X_1 \Rightarrow -3.6863 = -0.05X_1 \Rightarrow X_1 = \frac{3.6863}{0.05} \approx 73.73$$

Interpretation:

- The student needs to study about 73.7 hours to have an 80% chance of getting an A.

4 - Problem 4

4.0.1 (a)

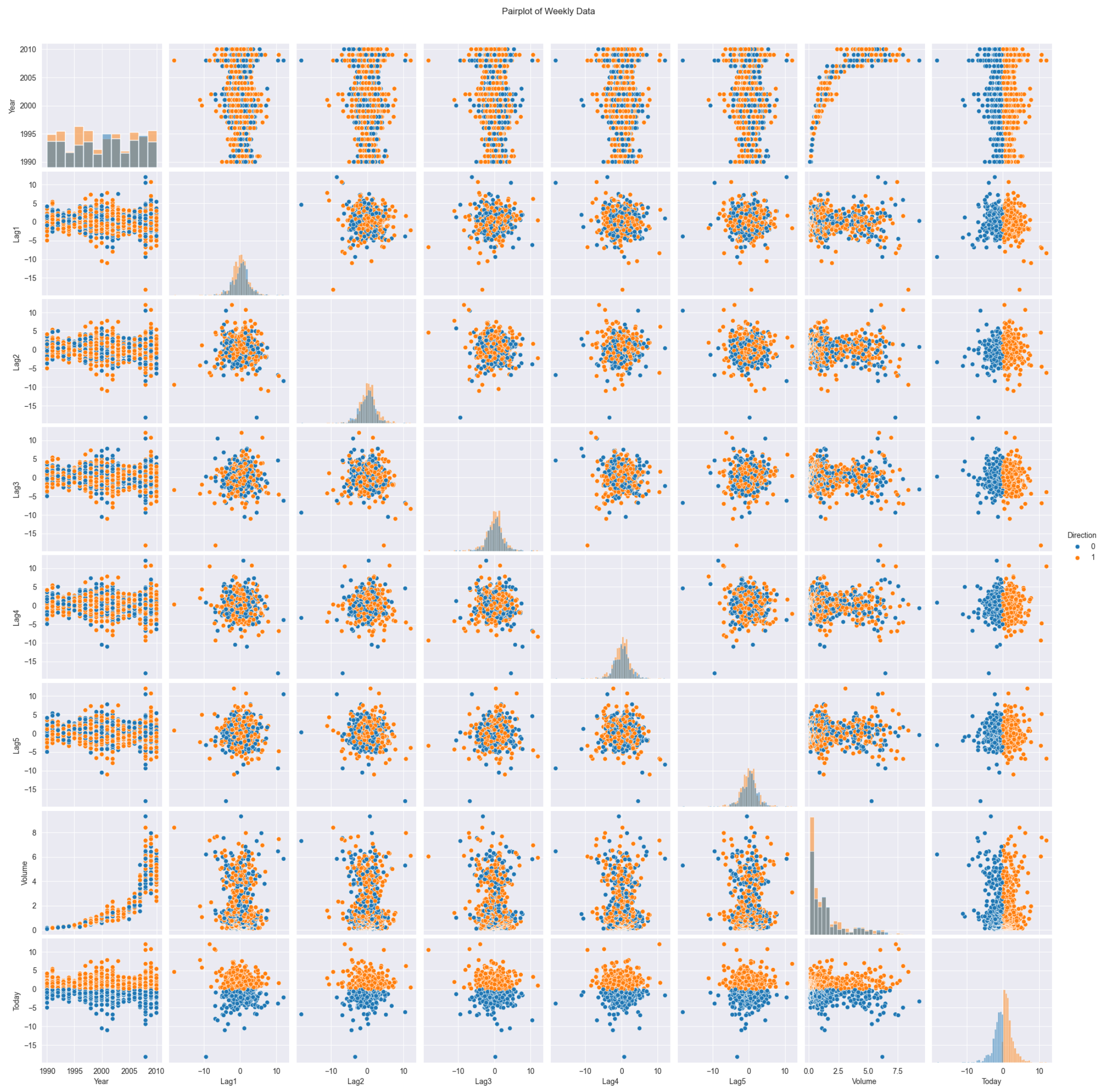
[ ]:
```
# Imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from ISLP import load_data

# Load data
Weekly = load_data('Weekly')

# Summary statistics
Weekly.describe()
```

[ ]:
```
              Year         Lag1         Lag2         Lag3         Lag4  \
count  1089.000000  1089.000000  1089.000000  1089.000000  1089.000000
mean   2000.048669     0.150585     0.151079     0.147205     0.145818
std       6.033182     2.357013     2.357254     2.360502     2.360279
min    1990.000000   -18.195000   -18.195000   -18.195000   -18.195000
25%    1995.000000    -1.154000    -1.154000    -1.158000    -1.158000
50%    2000.000000     0.241000     0.241000     0.241000     0.238000
75%    2005.000000     1.405000     1.409000     1.409000     1.409000
max    2010.000000    12.026000    12.026000    12.026000    12.026000

              Lag5       Volume        Today
count  1089.000000  1089.000000  1089.000000
mean      0.139893     1.574618     0.149899
std       2.361285     1.686636     2.356927
min     -18.195000     0.087465   -18.195000
25%      -1.166000     0.332022    -1.154000
50%       0.234000     1.002680     0.241000
75%       1.405000     2.053727     1.405000
max      12.026000     9.328214    12.026000
```

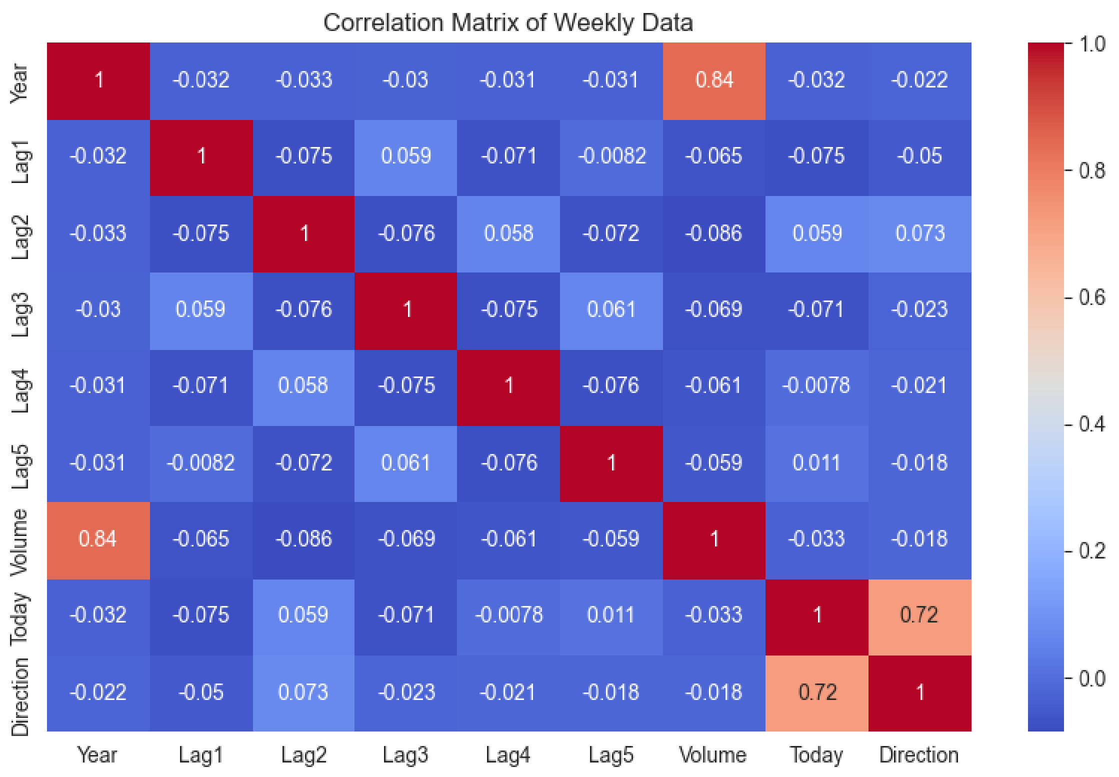
[27]:
```
# Preprocessing
Weekly['Direction'] = Weekly['Direction'].str.strip().map({'Up': 1, 'Down': 0})
Weekly.head()
```

[27]:
```
   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          0
1  1990 -0.270  0.816  1.572 -3.936 -0.229  0.148574 -2.576          0
2  1990 -2.576 -0.270  0.816  1.572 -3.936  0.159837  3.514          1
3  1990  3.514 -2.576 -0.270  0.816  1.572  0.161630  0.712          1
4  1990  0.712  3.514 -2.576 -0.270  0.816  0.153728  1.178          1
```

[28]:
```
# Pair plot
sns.pairplot(Weekly, hue='Direction', diag_kind='hist')

plt.suptitle("Pairplot of Weekly Data", y=1.02)
plt.show()
```



[29]:
```
# Heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(Weekly.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix of Weekly Data")
plt.show()
```



There is clearly a pattern between year and volume. In the pair plot, the volume is exponentially higher as year increases, especially after 2005. The heatmap confirms this trend, as there is a correlation coefficient of 0.84. Other than that, there is correlation with return today and direction, but this is due to the fact that direction is based on whether the return today is positive or negative. The lags and the return today seem to have an approximate normal distribution with $\mu \approx 0.14$ and $\sigma \approx 2.36$. The volume histogram also shows a right-skew, as there are many more data points for lower volume.

4.0.2 (b)

[30]:
```
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm

# Predictor variables
X = Weekly[['Lag1', 'Lag2', 'Lag3', 'Lag4', 'Lag5', 'Volume']]
y = Weekly['Direction']

# Add intercept and fit model
X_const = sm.add_constant(X)
logit_model = sm.Logit(y, X_const).fit()
logit_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.682441
         Iterations 4
```

[30]:

Dep. Variable:	Direction	**No. Observations:**	1089
Model:	Logit	**Df Residuals:**	1082
Method:	MLE	**Df Model:**	6
Date:	Fri, 25 Apr 2025	**Pseudo R-squ.:**	0.006580
Time:	11:52:16	**Log-Likelihood:**	-743.18
converged:	True	**LL-Null:**	-748.10
Covariance Type:	nonrobust	**LLR p-value:**	0.1313

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.2669	0.086	3.106	0.002	0.098	0.435
Lag1	-0.0413	0.026	-1.563	0.118	-0.093	0.010
Lag2	0.0584	0.027	2.175	0.030	0.006	0.111
Lag3	-0.0161	0.027	-0.602	0.547	-0.068	0.036
Lag4	-0.0278	0.026	-1.050	0.294	-0.080	0.024
Lag5	-0.0145	0.026	-0.549	0.583	-0.066	0.037
Volume	-0.0227	0.037	-0.616	0.538	-0.095	0.050

When $\alpha = 0.05$, we can see that only lag 2 is a significant predictor. We can see that lag 2 is the only predictor where 0 is not in the confidence interval.

4.0.3 (c)

```
[38]: from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
       ↪recall_score, f1_score

      # Predict
      y_pred = logit_model.predict(X_const) > 0.5
      cm = confusion_matrix(y, y_pred)
      cm_df = pd.DataFrame(cm,
                           index=['Actual Down', 'Actual Up'],
                           columns=['Predicted Down', 'Predicted Up'])
      acc = accuracy_score(y, y_pred)
      precision = precision_score(y, y_pred)
      recall = recall_score(y, y_pred)
      f1 = f1_score(y, y_pred)

      print(cm_df)
      print(f"\nAccuracy: {acc:.3f}")
      print(f"\nPrecision: {precision:.3f}")
      print(f"\nRecall: {recall:.3f}")
      print(f"\nF1: {f1:.3f}")
```

```
             Predicted Down  Predicted Up
Actual Down              54           430
Actual Up                48           557

Accuracy: 0.561

Precision: 0.564

Recall: 0.921

F1: 0.700
```

The confusion matrix shows: - True positives (Actual Up predicted as Up) - False negatives (Actual Up predicted as Down) - True negatives (Actual Down as predicted Down) - False positives (Actual Down predicted as Up)

Here are some helpful calculations:

$$\text{False Positive Rate} = \frac{\text{False Positive}}{\text{False Positive} + \text{True Negative}} = \frac{430}{430 + 54} = 0.888$$

$$\text{False Negative Rate} = \frac{\text{False Negative}}{\text{False Negative} + \text{True Positive}} = \frac{48}{48 + 557} \approx 0.079$$

The confusion matrix shows that our model is not that accurate as the F1 score comes in at 70%. F1 score is a better measure of accuracy because the number of total positives is way larger than the number of total negatives. We can also see that our model does well with finding most instances of positive days, as recall is 92.1%. However, the positive predictions have low accuracy, as the precision is only 56.4%. The False Positive Rate shows that 88.8% of the time, our model misrepresents down days as up days. This means that the model frequently predicts up when the actual direction is down. Type I errors are very common and seem problematic. On the other hand, the False Negative Rate shows that 7.9% of the time, our model misrepresents up days as down days. This shows that our model is mostly capturing up days correctly. The type II error rate is much lower than the type I error rate.

4.0.4 (d)

[40]:
```python
# Split data
train = Weekly[Weekly['Year'] <= 2008]
test = Weekly[Weekly['Year'] > 2008]

X_train = train[['Lag2']]
y_train = train['Direction']
X_test = test[['Lag2']]
y_test = test['Direction']

# Fit logistic regression
model_lag2 = LogisticRegression()
model_lag2.fit(X_train, y_train)
y_pred_test = model_lag2.predict(X_test)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred_test)
cm_df = pd.DataFrame(cm,
                     index=['Actual Down', 'Actual Up'],
                     columns=['Predicted Down', 'Predicted Up'])
acc = accuracy_score(y_test, y_pred_test)
precision = precision_score(y_test, y_pred_test)
recall = recall_score(y_test, y_pred_test)
f1 = f1_score(y_test, y_pred_test)

print(cm_df)
print(f"\nAccuracy: {acc:.3f}")
print(f"\nPrecision: {precision:.3f}")
print(f"\nRecall: {recall:.3f}")
print(f"\nF1: {f1:.3f}")
```

```
             Predicted Down  Predicted Up
Actual Down               9            34
Actual Up                 5            56

Accuracy: 0.625

Precision: 0.622

Recall: 0.918

F1: 0.742
```

The confusion matrix shows that our model is not that accurate as the accuracy comes in at 62.5%. This means that we are only predicting values correctly 62.5% of the time. We can also see that our model still does well with finding most instances of positive days, as recall is 91.8%. However, the positive predictions still have low accuracy, as the precision is only 62.2%.

4.0.5 (e)

```
[42]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
y_pred_lda = lda.predict(X_test)

cm = confusion_matrix(y_test, y_pred_lda)
cm_df = pd.DataFrame(cm,
                     index=['Actual Down', 'Actual Up'],
                     columns=['Predicted Down', 'Predicted Up'])
acc = accuracy_score(y_test, y_pred_lda)
precision = precision_score(y_test, y_pred_lda)
recall = recall_score(y_test, y_pred_lda)
f1 = f1_score(y_test, y_pred_lda)

print(cm_df)
print(f"\nAccuracy: {acc:.3f}")
print(f"\nPrecision: {precision:.3f}")
print(f"\nRecall: {recall:.3f}")
print(f"\nF1: {f1:.3f}")
```

```
             Predicted Down  Predicted Up
Actual Down               9            34
Actual Up                 5            56

Accuracy: 0.625

Precision: 0.622

Recall: 0.918

F1: 0.742
```

We do not see an improvement with LDA, as we get identical statistics and confusion matrix.

4.0.6 (f)

```
[43]: from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
y_pred_qda = qda.predict(X_test)

cm = confusion_matrix(y_test, y_pred_qda)
cm_df = pd.DataFrame(cm,
                     index=['Actual Down', 'Actual Up'],
                     columns=['Predicted Down', 'Predicted Up'])
acc = accuracy_score(y_test, y_pred_qda)
precision = precision_score(y_test, y_pred_qda)
recall = recall_score(y_test, y_pred_qda)
f1 = f1_score(y_test, y_pred_qda)

print(cm_df)
print(f"\nAccuracy: {acc:.3f}")
print(f"\nPrecision: {precision:.3f}")
print(f"\nRecall: {recall:.3f}")
print(f"\nF1: {f1:.3f}")
```

```
             Predicted Down  Predicted Up
Actual Down               0            43
Actual Up                 0            61

Accuracy: 0.587

Precision: 0.587

Recall: 1.000

F1: 0.739
```

This model is worse, as the accuracy is lower, meaning that it predicts the values correctly less of the time. It doesn't address the issue of low precision.

4.0.7 (g)

The Logistic Regression or LDA both provide the best results from the models we compared on this data. But frankly, the accuracies and precisions of both models are still garbage, meaning that there could be an underlying relationship in the data that we aren't capturing, resulting in low precision. This is likely, especially since day movements are usually determined by way more factors than the movement 2 days ago. We may need to investigate more possible predictors to get more accurate results.

5 - Problem 5

In k-fold cross-validation:

- The dataset is randomly divided into k equal (or nearly equal) parts, or **folds**.
- The model is trained on $k - 1$ folds and tested on the remaining fold.
- This process is repeated k times, each time with a different fold as the test set.
- The average test error across all k iterations is reported as the cross-validation estimate.

For example, here are the steps for 5-fold cross-validation:

- Split data into 5 parts (say A, B, C, D, E).
- Train on A+B+C+D, test on E
- Train on A+B+C+E, test on D
- ... and so on, until each fold has been used as test data once.
- Then we take the average the MSE of all tests as our final MSE.

1. k-Fold CV vs. Validation Set Approach:

	k-Fold CV	Validation Set
Stability	More stable error estimate	Can be higher variance
Data usage	All data used for training & testing	Only part of data used for training
Computation	More expensive (train k models)	Very fast

- **Advantage of k-Fold:** Ensures all data is included in the test dataset at least once and gives more reliable estimates. Compared to Validation Set, where you randomly select data to be in the test dataset, which may not be an accurate representation of test dataset, especially when the data is imbalanced (unequal representations of classes).
- **Disadvantage of k-Fold:** Slower due to multiple model training runs. Compared to Validation Set, which has only one iteration.

2. k-Fold CV vs. LOOCV (Leave-One-Out):

- LOOCV is a special case of k-fold CV with $k = n$ (number of observations).
- Each fold contains only 1 observation as the test set.

	k-Fold CV	LOOCV
Bias	Slightly higher bias	Nearly unbiased
Variance	Lower variance	High variance
Computation	Reasonable Speed (i.e. $k = 5$)	Very slow (train n models)

- **Advantage of k-Fold:** Generally, computational cost is much lower while while maintaining a good variance-bias tradeoff.
- **Advantage of LOOCV:** Reduces bias, but comes with much higher computational cost and variance due to the test set being just one data point.

Conclusion: In practice, k-Fold CV is generally best, as it is a fair tradeoff between lower MSE and lower computational cost. We usually select $k = 5$ or $k = 10$.

6 - Problem 6

We are assuming that we have a complex model with no built in formula to calculate variance. In this case, the best way to estimate variance is using Bootstrapping, which:

- Repeatedly resamples the training data with replacement
- Fits the model on each bootstrap sample
- Predicts $\hat{Y}$ for the same X each time
- Computes the standard deviation of the resulting predictions

```
# Pseudocode
predictions = []
for b in range(B):
    boot_sample = resample(data)
    model.fit(boot_sample)
    y_hat = model.predict(X)
    predictions.append(y_hat)

std_dev = np.std(predictions)
```

With bootstrap, we are assuming that our data is a good representation of the population. Resampling it with replacement simulates the process of drawing different training sets from the population. This helps us understand how much prediction variability there can be if we had different samples.