Project Check-In: Analysis/Questions/Road Map

Molly Scheitler

Initial Analysis - Univariate

I performed all steps for all columns (statistics and visuals for unemployment, ^GSPC, ^IXIC, ^DJI, Interest_Rate, and Inflation_Rate)

Unemployment_Rate Column

```
# Find all unique values in the Unemployment_Rate column

print(final_dataset["Unemployment_Rate"].unique())
```

```
[ 7.3  7.4  7.6  7.8  7.7  7.1  7.   6.9  6.8  6.7  6.6  6.5  6.4  6.1
  6.   5.9  5.8  5.6  5.5  5.4  5.7  5.3  5.1  5.2  4.9  5.   4.8  4.7
  4.6  4.3  4.4  4.5  4.2  4.1  4.   3.8  3.9  6.3  6.2  8.3  8.7  9.
  9.4  9.5  9.6  9.8 10.   9.9  9.3  9.1  8.8  8.6  8.5  8.2  8.1  7.9
  8.   7.5  7.2  3.7  3.6  3.5 14.8 13.2 11.  10.2  8.4  3.4]
```

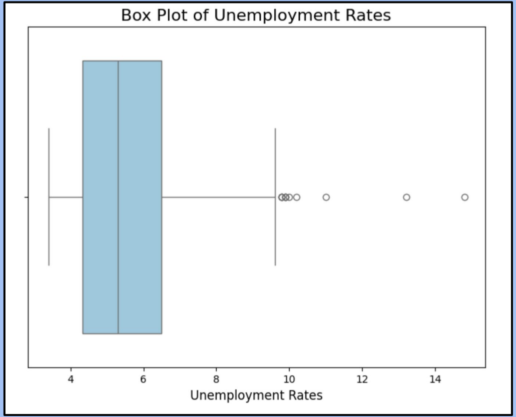
```
# Use the .describe() function to find the descriptive statistics of the

print("Descriptive Statistics of Unemployment_Rate column:\n")

descriptive_statistics = final_dataset["Unemployment_Rate"].describe()

display(descriptive_statistics)
```

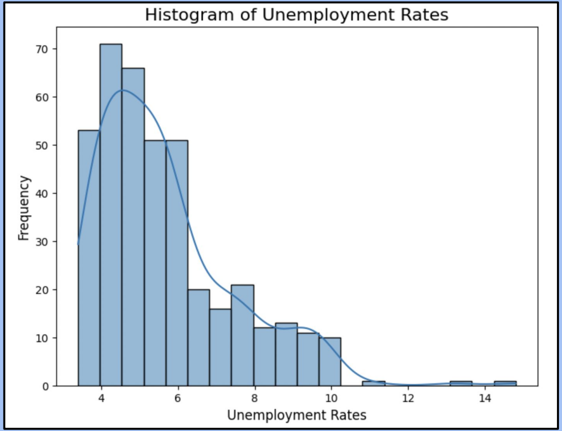
```
Descriptive Statistics of Unemployment_Rate column:

count    398.000000
mean       5.675126
std        1.787307
min        3.400000
25%        4.325000
50%        5.300000
75%        6.500000
max       14.800000
Name: Unemployment_Rate, dtype: float64
```

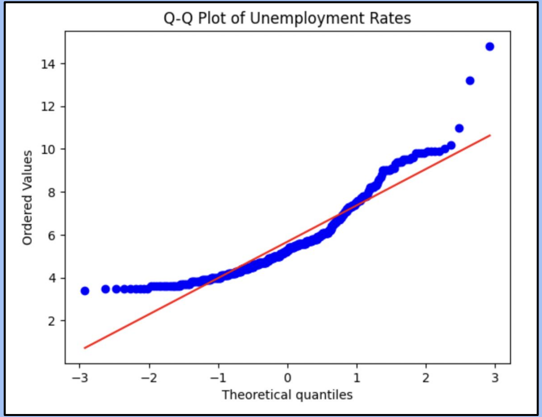
```
# Complete the Shapiro-Wilk test for normality

# Shapiro-Wilk test for normality
stat, p_value = stats.shapiro(column)

# Display the results
print(f"Unemployment Rate Shapiro-Wilk Test Statistic: {stat}")
print(f"Unemployment Rate P-Value: {p_value}")

# Interpretation
alpha = 0.05  # Significance level
if p_value > alpha:
    print("The data follows a normal distribution (fail to reject H0).")
else:
    print("The data does not follow a normal distribution (reject H0).")
```

```
Unemployment Rate Shapiro-Wilk Test Statistic: 0.891335666179657
Unemployment Rate P-Value: 3.409993974083631e-16
The data does not follow a normal distribution (reject H0).
```

I concluded that most columns have outliers and are heavily right-skewed. I need to work on removing outliers to get normal distributions.







Initial Analysis - Bivariate

Unemployment_Rate and ^GSPC Columns

```python
# Calculate Pearson correlation between 'Unemployment_Rate' and '^GSPC'

# Pearson correlation
pearson_corr = final_dataset["Unemployment_Rate"].corr(final_dataset["^GSPC"], method="pearson")

# Display the result
print(f"Pearson Correlation: {pearson_corr:.4f}")

# Interpretation
if abs(pearson_corr) < 0.3:
    print("Interpretation: No or weak linear correlation.")
elif 0.3 <= abs(pearson_corr) < 0.7:
    print("Interpretation: Moderate linear correlation.")
else:
    print("Interpretation: Strong linear correlation.")
```

```
Pearson Correlation: -0.3759
Interpretation: Moderate linear correlation.
```

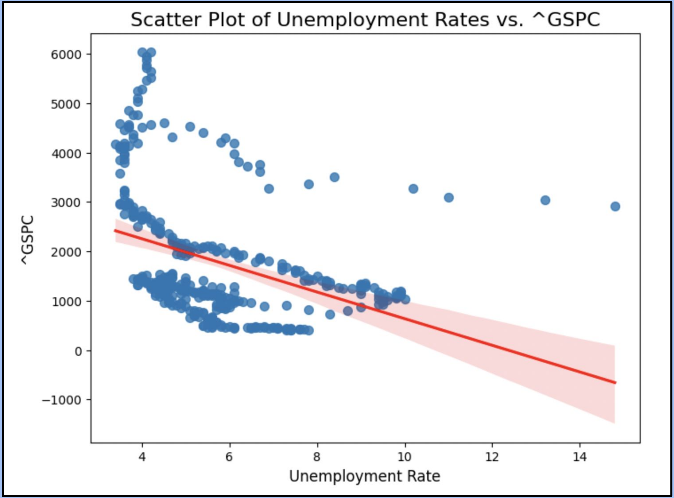
I performed all these steps with unemployment and every other column (unemployment and ^IXIC for example)

I performed these steps with interest and every other column as well (interest and inflation for example)

I concluded that all stocks, interest, and inflation have a negative correlation with unemployment. All stocks have a negative correlation with interest. Inflation has a positive correlation with interest.

```python
# Perform the Pearson correlation test

# Pearson correlation test
corr_stat, p_value = pearsonr(final_dataset["Unemployment_Rate"], final_dataset["^GSPC"])

# Display the correlation statistic and p-value
print(f"Pearson Correlation Coefficient: {corr_stat:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpretation based on p-value
alpha = 0.05  # significance level

if p_value > alpha:
    print("Fail to reject H0: No significant correlation (correlation = 0).")
else:
    print("Reject H0: Significant correlation exists (correlation != 0).")
```

```
Pearson Correlation Coefficient: -0.3759
P-value: 0.0000
Reject H0: Significant correlation exists (correlation != 0).
```



Initial Analysis - Hypothesis Tests

Two-Sample T-Tests

Comparing Unemployment Rates From the 1990s to the 2000s

```python
# Create a new column for decade
final_dataset['Decade'] = (final_dataset['Date'].dt.year // 10) * 10

# Filter for two decades (1990s and 2000s)
unemployment_1990s = final_dataset[final_dataset['Decade'] == 1990]['Unemployment_Rate']
unemployment_2000s = final_dataset[final_dataset['Decade'] == 2000]['Unemployment_Rate']

# Run a two-sample t-test
t_stat, p_value = ttest_ind(unemployment_1990s, unemployment_2000s, equal_var=False)

# Show results for t-statistic and p-value
print("T-Statistic:", t_stat)
print("P-value:", p_value)

# Interpret the results according to alpha
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: Average unemployment is significantly different between the 1990s and 2000s.")
else:
    print("Fail to reject the null hypothesis: No significant difference in average unemployment the 1990s and 2000s.")
```

```
T-Statistic: 0.5968223622548725
P-value: 0.5512589310147289
Fail to reject the null hypothesis: No significant difference in average unemployment the 1990s and 2000s.
```

I split up all my “Date” data into decades and compared the decades (1990s-2000s, 2000s-2010s, and 2010s-2020s)

I concluded that from the 1990s to the 2000s, there is not a significant difference in the unemployment rates. From the 2000s to the 2010s, there is a significant difference in unemployment. From the 2010s to the 2020s, there is a significant difference in unemployment.

Comparing Unemployment Rates From the 2000s to the 2010s

```python
# Create a new column for decade
final_dataset['Decade'] = (final_dataset['Date'].dt.year // 10) * 10

# Filter for two decades (2000s and 2010s)
unemployment_2000s = final_dataset[final_dataset['Decade'] == 2000]['Unemployment_Rate']
unemployment_2010s = final_dataset[final_dataset['Decade'] == 2010]['Unemployment_Rate']

# Run a two-sample t-test
t_stat, p_value = ttest_ind(unemployment_2000s, unemployment_2010s, equal_var=False)

# Show results for t-statistic and p-value
print("T-Statistic:", t_stat)
print("P-value:", p_value)

# Interpret the results according to alpha
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: Average unemployment is significantly different between the 2000s and 2010s.")
else:
    print("Fail to reject the null hypothesis: No significant difference in average unemployment the 2000s and 2010s.")
```

```
T-Statistic: -2.9539201415970164
P-value: 0.0034895472773083087
Reject the null hypothesis: Average unemployment is significantly different between the 2000s and 2010s.
```

Initial Analysis - Machine Learning

Linear Regression Model

```python
# Find the target and predictor variables

# Predictor variables
X = final_dataset.drop(['Unemployment_Rate', 'Date'], axis=1)

# Target variable
y = final_dataset['Unemployment_Rate'].values
```

```python
# Linear Regression Model

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42)

Lin_Reg = LinearRegression()

# Fit the training data to the Linear Regression
Lin_Reg.fit(X_train, y_train)

# Get the predictions
y_pred = Lin_Reg.predict(X_test)

# Prediction is a straight line: y = ax + b
# R^2 score - model not working well - very low - not good for making predictions on this pandas DataFrame
print(f'Linear Regression R^2: {Lin_Reg.score(X_test, y_test)}')
```

```
Linear Regression R^2: 0.6783667055712629
```

```python
# Linear Regression y-intercept

intercept = Lin_Reg.intercept_
print(f'Linear Regression intercept: {intercept}')
```

```
Linear Regression intercept: -318.11978205660967
```

```python
# Coefficients of the Linear Regression

# Many coefficients because each column has a coefficient
print(f'Linear Regression coefficients: {Lin_Reg.coef_}')
```

```
Linear Regression coefficients: [-0.00268584  0.00103966 -0.00032919 -0.2413405  -0.19965323  0.1647241 ]
```

```python
# Step 13 - Print the columns of final_dataset pandas DataFrame

print(f'Allstate columns: {final_dataset.columns}')
```

```
Allstate columns: Index(['Date', 'Unemployment_Rate', '^GSPC', '^IXIC', '^DJI', 'Interest_Rate',
       'Inflation_Rate', 'Decade'],
      dtype='object')
```

Support Vector Classifier Model

```python
# Support Vector Classifier Model

svm = SVC()

# Fit the classifier to the training data
svm.fit(X_train, y_train)

# Get the predictions
y_pred = svm.predict(X_test)

# Print Accuracy Score
print(f'SVC Accuracy: {svm.score(X_test, y_test)}')

# Print F1 Score
print(f"SVC F1 Score: {f1_score(y_test, y_pred, average='weighted'):.4f}")
```

```
SVC Accuracy: 0.5125
SVC F1 Score: 0.4614
```

Logistic Regression Model

```python
# Logistic Regression Model

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Ensuring no warnings appear in output with this classifier
final_log = LogisticRegression(solver='liblinear', max_iter=4000, class_weight='balanced')

# Fit the training data to the Logistic Regression model
final_log.fit(X_train,y_train)

# Get the predictions
y_pred = final_log.predict(X_test)

# Print the Accuracy Score
print(f"Logistic Regression Accuracy: {final_log.score(X_test, y_test)}")

# Print the F1 Score
print(f"Logistic Regression F1 Score: {f1_score(y_test, y_pred, average='weighted'):.4f}")
```

```
Logistic Regression Accuracy: 0.6375
Logistic Regression F1 Score: 0.6240
```

Initial Analysis - Machine Learning

K-Nearest Neighbors Classifier Model

```
# KNN Classifier Model

knn = KNeighborsClassifier(n_neighbors=7)

# Fit the training data to the model
knn.fit(X_train, y_train)

# Get the predictions
y_pred = knn.predict(X_test)

# Confusion Matrix
confusion_matrix(y_test, y_pred)
```

```
array([[16,  3,  0,  1],
       [ 7, 17,  0,  0],
       [ 2,  1, 13,  0],
       [ 0,  0,  1, 19]])
```

```
# Classification Report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        High       0.64      0.80      0.71        20
         Low       0.81      0.71      0.76        24
   Very High       0.93      0.81      0.87        16
    Very Low       0.95      0.95      0.95        20

    accuracy                           0.81        80
   macro avg       0.83      0.82      0.82        80
weighted avg       0.83      0.81      0.82        80
```

```
# Print results

# Print Accuracy Score
print(f'KNN Accuracy: {knn.score(X_test, y_test)}')

# Print F1 Score
print(f"KNN F1 Score: {f1_score(y_test, y_pred, average='weighted'):.4f}")
```

```
KNN Accuracy: 0.8125
KNN F1 Score: 0.8153
```

Decision Tree Classifier Model

```
# Convert the Unemployment_Rate column to Unemployment_Group indicating the level of unemployment

final_dataset["Unemployment_Group"] = pd.qcut(final_dataset["Unemployment_Rate"], q=4, labels=["Very Low", "Low", "High", "Very High"])
```

```
# Find the target and predictor variables

X = final_dataset.drop(columns=["Unemployment_Rate", "Unemployment_Group", "Date", "Decade"])
y = final_dataset["Unemployment_Group"]
```

```
# Decision Tree Classifier Model

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Dec_tree = DecisionTreeClassifier(random_state=42)

# Fit the classifier to the training data
Dec_tree.fit(X_train, y_train)

# Get the predictions
y_pred = Dec_tree.predict(X_test)

# Print Accuracy Score
print(f'Decision Tree Accuracy: {Dec_tree.score(X_test,y_test)}')

# Print F1_score
print(f"Decision Tree F1 Score: {f1_score(y_test, y_pred, average='weighted'):.4f}")
```

```
Decision Tree Accuracy: 0.8875
Decision Tree F1 Score: 0.8848
```

All my variables are continuous, so I had to convert unemployment to a categorical variable to perform these techniques.

<u>Linear Regression:</u> Pretty strong fit with r^2 = .6783
<u>SVC:</u> Not a strong fit with accuracy = .5125
<u>Logistic Regression:</u> Not a great fit with accuracy = .6375
<u>KNN:</u> Strong fit with accuracy = .8125
<u>Decision Tree:</u> Strong fit with accuracy = .8875.
<u>In conclusion, the Decision Tree Classifier is the best fit for my data.</u>

Questions Answered with Initial Analysis

1) **How does the unemployment rate correlate with interest rates over time?**
   - Negative correlation - as unemployment rates rise, the interest rates tend to decrease.
2) **How does the unemployment rate correlate with inflation rates over time?**
   - Negative correlation - as unemployment rates rise, inflation rates tend to decrease.
3) **How does the unemployment rate correlate with major stock market indices (S&P 500, Nasdaq Composite, and Dow Jones) over time?**
   - S&P: negative correlation - as unemployment rates rise, the stock price tends to decrease.
   - Nasdaq: negative correlation - as unemployment rates rise, the stock price tends to decrease.
   - Dow Jones: negative correlation - as unemployment rates rise, the stock price tends to decrease.
4) **How do stock market trends behave during periods of high vs. low unemployment?**
   - All three stock market indices have a negative correlation with unemployment rates. As unemployment increases, index values tend to decrease.
5) **How do interest rates correlate with stock indices and inflation rates?**
   - S&P: negative correlation - as interest rates rise, the stock price tends to decrease.
   - Nasdaq: negative correlation - as interest rates rise, the stock price tends to decrease.
   - Dow Jones: negative correlation - as interest rates rise, the stock price tends to decrease.
   - Inflation Rates: positive correlation - as interest rates rise, the inflation rates tend to increase as well.

Remaining Questions

1) Do I remove outliers? How do I know when and how many to remove?
2) Are there any other areas I should be exploring specifically with this data?
3) Should I be incorporating the “Date” column into my analysis and visualizations more?
4) How do I organize and display all final project pieces?

Road Map

What to accomplish by the end of the project:

1) Perform time series analysis
2) Use time series data to forecast future data
3) Incorporate high dimensional visualizations
4) Incorporate visualizations involving one categorical variable ("Date" in my dataset) and other continuous variables

How to answer remaining questions:

1) **Which stock market index (S&P 500, Nasdaq Composite, Dow Jones) is most sensitive to changes in unemployment rates?** I can place all regression lines from the scatter plots I have made (Unemployment v S&P, Unemployment v Nasdaq, etc) into one chart and compare the angles of the lines to find the answer. I can visually see the difference in the lines and decide sensitivity from there. I must have different colors for each stock and a legend for user interpretation.
2) **Does the stock market decline before or after an increase in unemployment rates?** I can use a line chart to visualize the unemployment rate across time to determine the dates that the rate rises or declines. When I know the specific date I am looking at, I can plot the stock market indices and the unemployment rate together in a line chart to draw conclusions from the date I chose to look into. I can utilize a time series plot.
3) **Does a rising inflation rate weaken the stock market more than an increasing unemployment rate?** I can use a histogram with scatter points to plot out market value and inflation as well as market value and unemployment. The distance of the points from the regression line will tell me which is more correlated. I can use a correlation histogram or a scatter matrix.
4) **How long does it take for the stock market to recover after a major spike in unemployment?** I must set a point that indicates "recovery" in the stock market which is up to my discretion. For example, the recovery value could be the mean stock market value or the last top peak in the chart. Then, I can chart out unemployment and the stock market to indicate how long recovery takes.