Assignment 6: Jump Predition

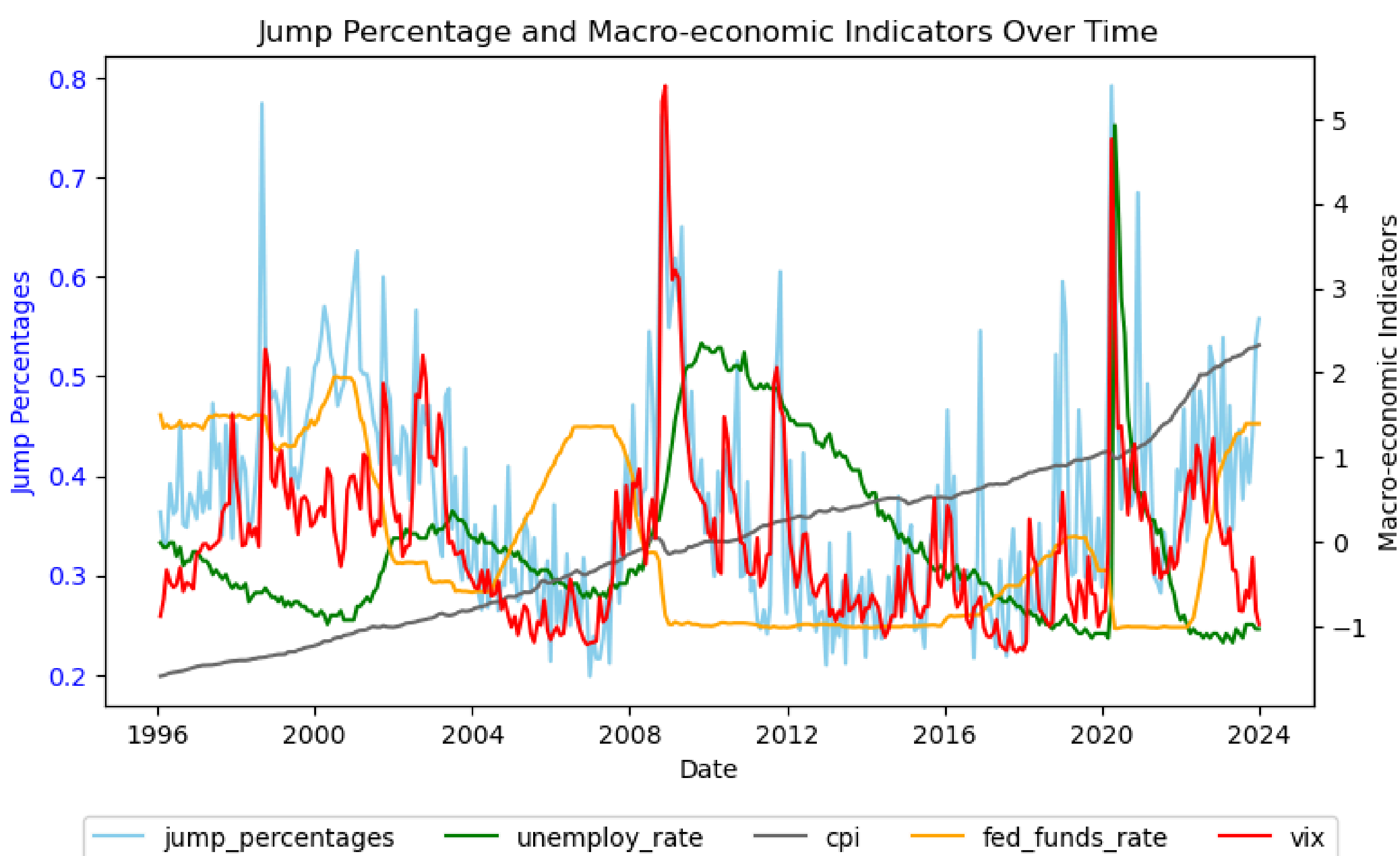
Name: Shiqi Hu
GTID: 904061372

Data Construction and Analysis

0. Data Import and Sample Firm Selection

Step 1: Construct a categorical outcome variable

Step 2: Plot the percentage of jumps over time along with the three macro-economic indicators from FRED



What do you observe?- Can any of these indicators serve as a precursor for upcoming higher likelihood of jumps?

Time series observations:

- The jump percentages appear highly volatile, with frequent spikes.
- There are noticeable spikes around 2008 (financial crisis) and 2020 (COVID-19 pandemic), where jump percentages, VIX, and unemployment rate all show significant increases.

Relationship with macro indicators:

- Jump percentages has very strong positive correlation (0.72) with VIX, indicating that periods of high market volatility are strongly associated with a higher likelihood of jump events.
- However, in comparison, jump percentages has weak to moderate correlations with other indicators.
  - Fed funds rate shows a weak positive correlation (0.24) with jump percentages.
  - CPI (Consumer Price Index) has a weak negative correlation (-0.13) with jump percentages.
  - Unemployment rate has a very weak negative correlation (-0.042) with jump percentages.

Precursors analysis

- while all these indicators provide some insight, the VIX stands out as the most promising precursor for upcoming higher likelihood of jumps. However, a holistic approach considering multiple indicators, especially during periods of economic stress or uncertainty, would likely provide the most robust predictive power for jump events.

Step 3: Come up with a list of possible covariates that should matter for jump prediction.

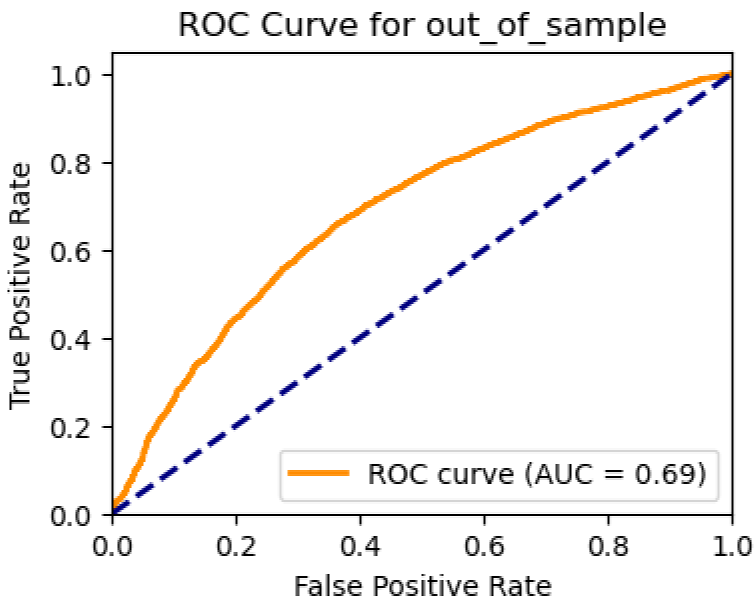
List of possible covariates:

- Predicted neural beta
- Industry (dummy)
- Lagging 12 month price, return, market return, month volume, market capitalization, annualized volatility, USREC, VIX, unemployment rate, cpi, and fed fund rate.

Sample 10 company each industry each year, and get the whole year data for the company selected in that year.

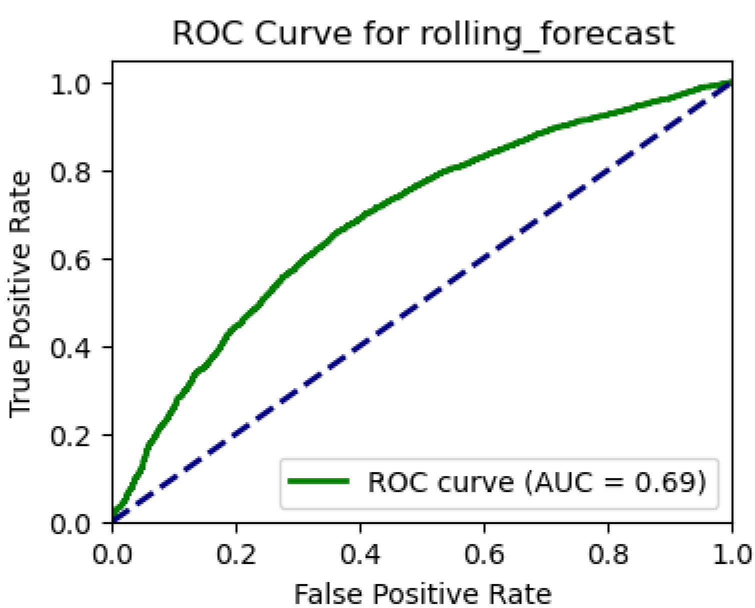
Using 2018–2023 as the out of sample period

```
Full Out-of-Sample - AUC: 0.6875
Full Out-of-Sample - KS Statistic: 0.2958
```



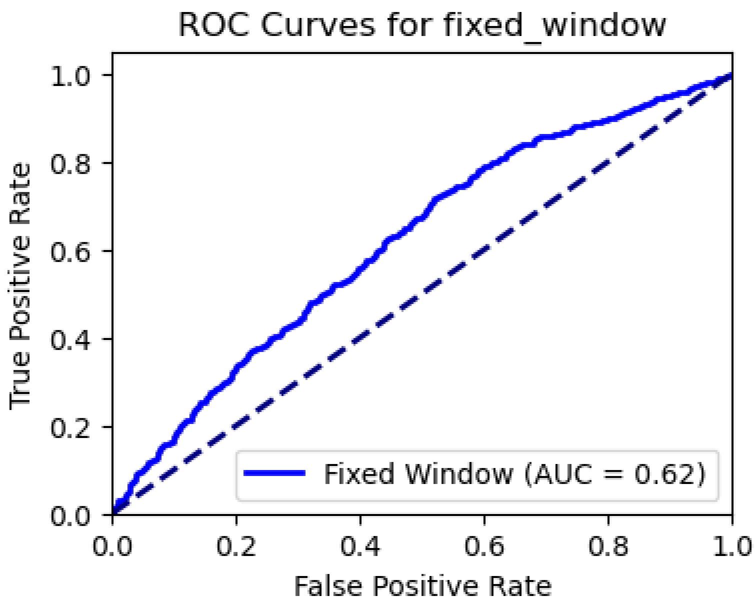
Doing a rolling out of sample estimation

```
Rolling Forecast - AUC: 0.6875
Rolling Forecast - KS Statistic: 0.2958
```



Using a fixed window

```
Fixed Window Forecast - AUC: 0.6204
Fixed Window Forecast - KS Statistic: 0.1967
```

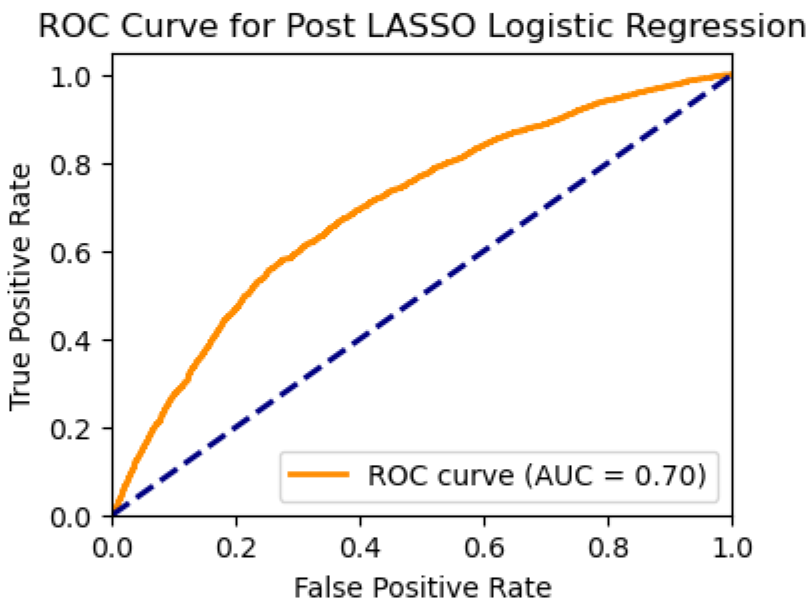


Contrast the results:

- The Full Out-of-Sample and Rolling Forecast methods show identical performance, with both having an AUC of 0.6875 and KS statistic of 0.2958. This suggests these approaches yield very similar predictive power.
- The Fixed Window Forecast method performs noticeably worse, suggesting it has significantly lower ability to discriminate between classes compared to using the full sample or rolling forecast approach.
- Therefore, from now on, I choose to use the full out-of-sample method as it has the best performance.

Model 2: LASSO Logistic regression

```
Best λ for LASSO: 0.002486421189550464
Post LASSO - AUC: 0.6977
Post LASSO - KS Statistic: 0.3047
```

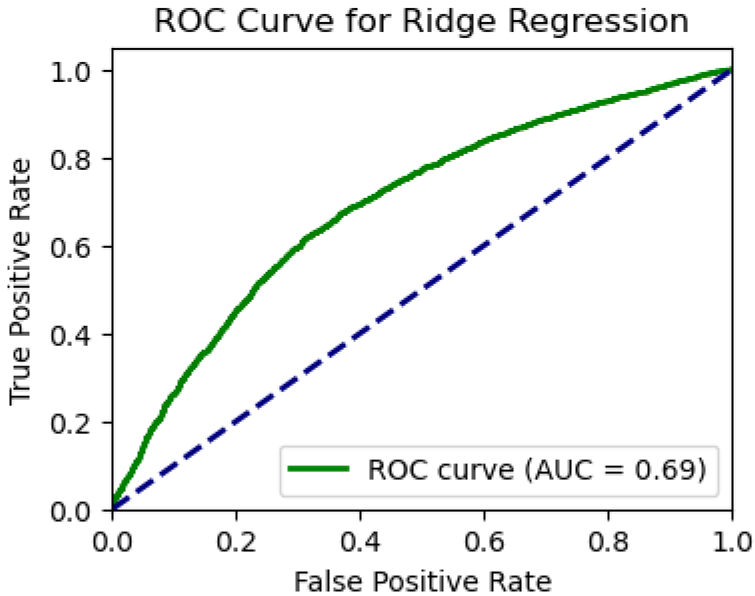


Compare model performance of the Post LASSO Logistic regression model with the Logistic regression model built in Model-1. Does it perform better?

- The Post LASSO model shows a slight improvement in AUC, indicating a marginal enhancement in the model's ability to distinguish between classes.
- Also, the KS statistic shows a small improvement. This suggests a slightly better separation between the positive and negative classes.
- The LASSO technique selected specific features, potentially reducing model complexity while maintaining or slightly improving performance.

Model 3: Ridge Logistic regression

```
Best λ for Ridge: 10.0
Ridge - AUC: 0.6904
Ridge - KS Statistic: 0.3034
```



Compare model performance of the Ridge Regression model with the Logistic regression model built in Model-1.

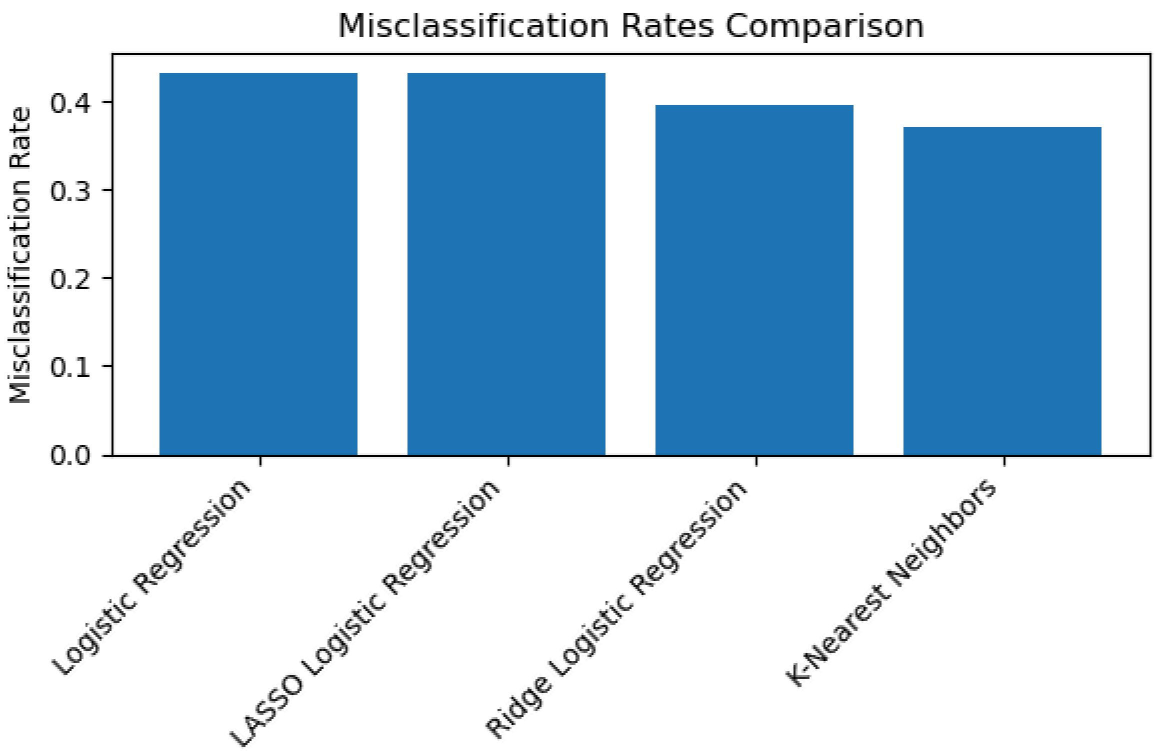
- The Ridge Regression model shows a slight improvement in AUC. This indicates a marginal enhancement in the model's discriminative ability.
- The KS statistic for the Ridge model is higher than that of the logistic model, suggesting the Ridge model achieves a slightly better separation between positive and negative classes.
- The Ridge Regression likely reduced overfitting compared to the standard logistic regression. A small but noticeable improvement in both AUC and KS statistic indicates better generalization to out-of-sample data.

Model- 4: K-Nearest Neighbor (KNN)

```
Best K: 30
Validation Misclassification Rate: 0.2976
Test Misclassification Rate (KNN): 0.3697

Misclassification Rates:
Logistic Regression: 0.4323
LASSO Logistic Regression: 0.4323
Ridge Logistic Regression: 0.3947
K-Nearest Neighbors: 0.3697

The model with the lowest misclassification rate is: K-Nearest Neighbors
```



Which one has the lowest misclassification rate?

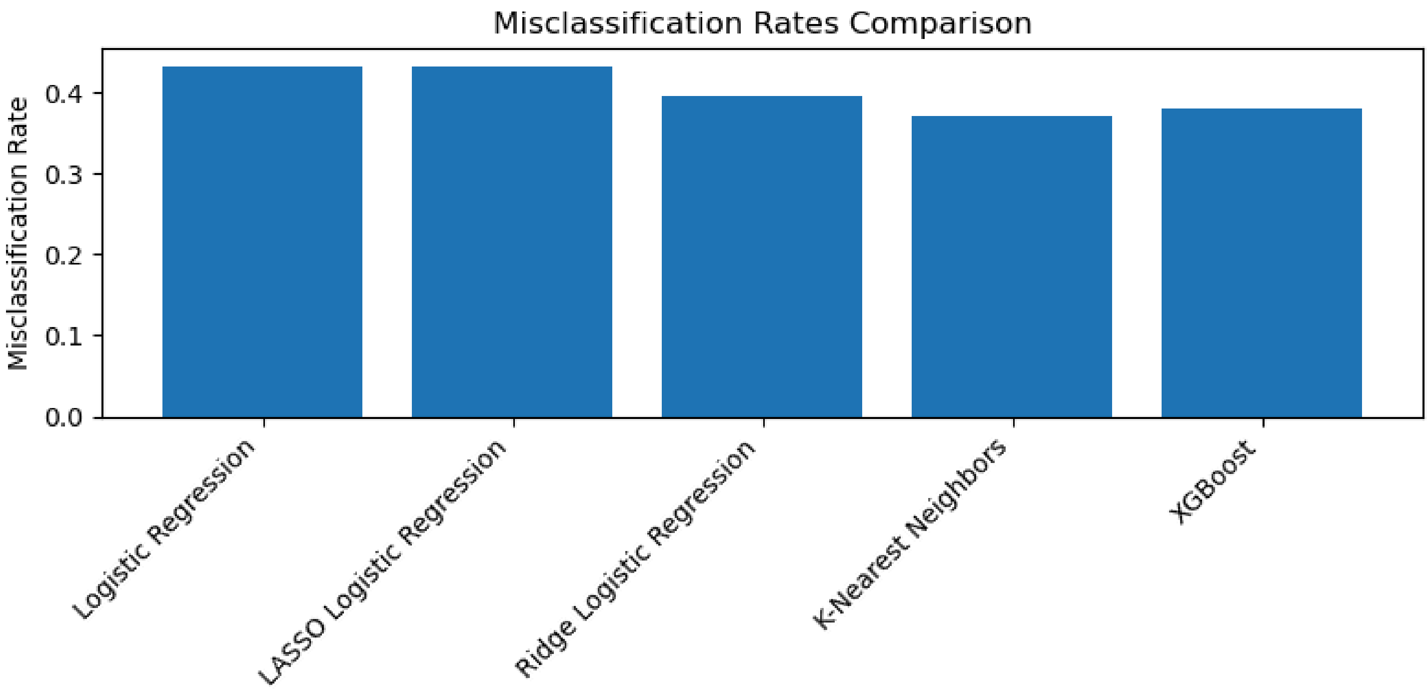
- The K-Nearest Neighbors model has the lowest misclassification rate at 0.3697. This is significantly lower than the Logistic Regression and LASSO Logistic Regression models, which all have misclassification rates above 0.43.
- Unlike the regression-based models, KNN adapts to local patterns in the feature space, which may be particularly effective for this dataset where linear decision boundaries are less optimal.

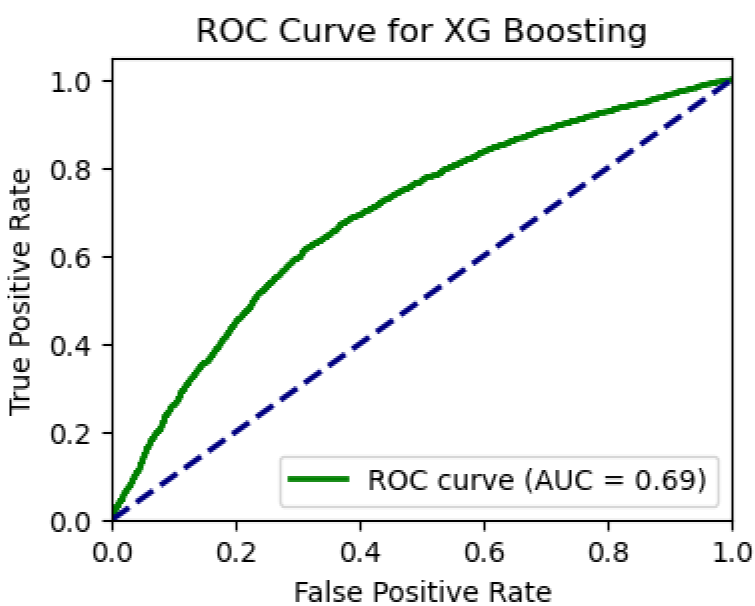
Models- 5: Pick ONE of the Gradient Boosted Tree Frameworks (XGBOOST)

```
Best parameters: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 100}
Validation Misclassification Rate: 0.3000
Test Misclassification Rate (XGBoost): 0.3792

Misclassification Rates:
Logistic Regression: 0.4323
LASSO Logistic Regression: 0.4323
Ridge Logistic Regression: 0.3947
K-Nearest Neighbors: 0.3697
XGBoost: 0.3792

The model with the lowest misclassification rate is: K-Nearest Neighbors
```





Compare misclassification rate of XGBoost algorithm with models built in earlier tasks. Which one has the lowest misclassification rate?

- The XGBoost algorithm has the misclassification rate at 0.3792, outperforming all other models except K-Nearest Neighbors. The KNN algorithm has the lowest misclassification rate at 0.3697, slightly outperforming XGBoost which has a rate of 0.3792.
- XGBoost performs better compare with most models because it's designed to learn complex patterns in data that simpler models might miss. It combines many decision trees and improves them step-by-step, allowing it to capture subtle relationships between features and make more accurate predictions.

- K-Nearest Neighbors likely performs best because it can capture complex local patterns without making strong assumptions about the overall data distribution.

Model 6. ArtificialNeuralNetwork(ANN)for3-classclassification

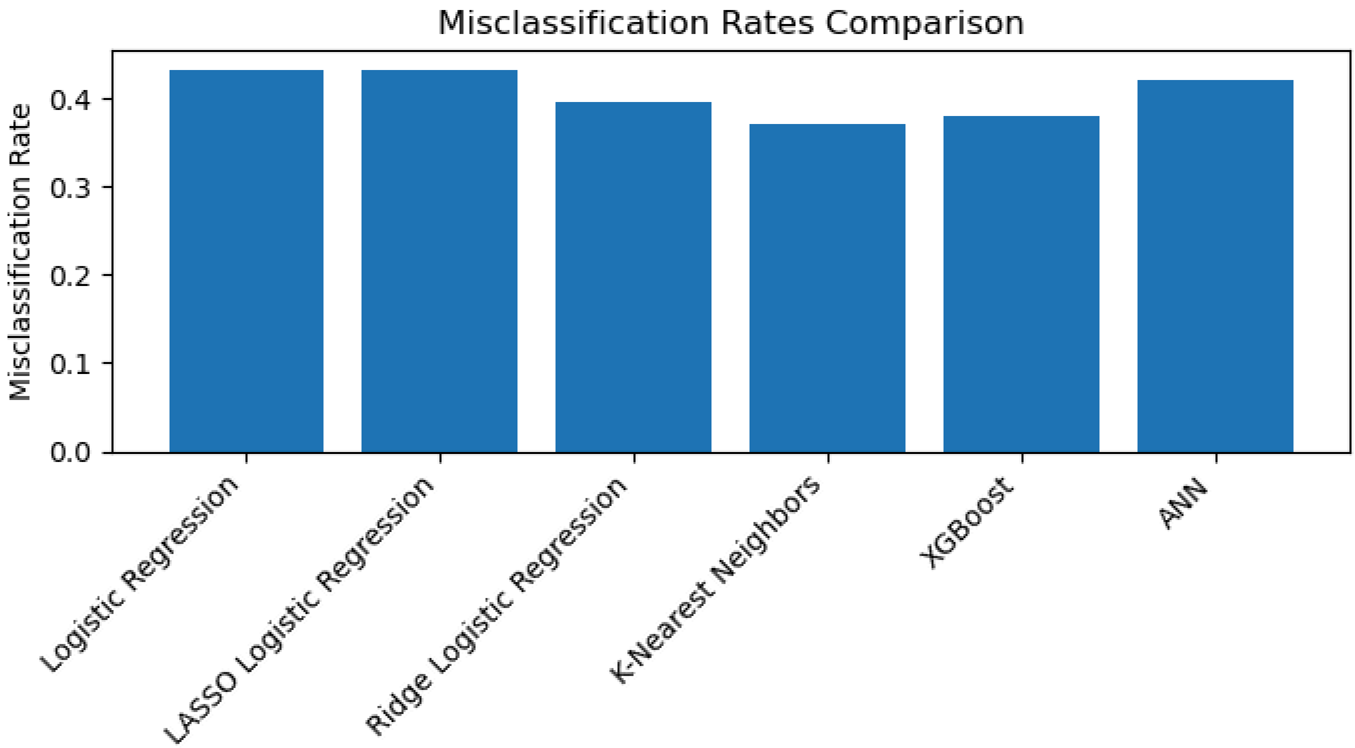
```
Epoch [10/50], Loss: 0.9447
Epoch [20/50], Loss: 0.6757
Epoch [30/50], Loss: 0.6649
Epoch [40/50], Loss: 0.7337
Epoch [50/50], Loss: 0.7746
ANN Misclassification Rate: 0.4203

Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.83      0.73      3966
           1       0.55      0.20      0.29      2553
           2       0.00      0.00      0.00         0

    accuracy                           0.58      6519
   macro avg       0.40      0.34      0.34      6519
weighted avg       0.61      0.58      0.56      6519

Misclassification Rates:
Logistic Regression: 0.4323
LASSO Logistic Regression: 0.4323
Ridge Logistic Regression: 0.3947
K-Nearest Neighbors: 0.3697
XGBoost: 0.3792
ANN: 0.4203
```



Compare the misclassification rate of MLP algorithm with models built in earlier tasks. Which one has the lowest misclassification rate?

- ANN underperforms compared with Ridge Logistic Regression, K-Nearest Neighbors and XGBoost algorithm but outperforms compared with Logistic Regression and LASSO Logistic Regression in misclassification rate.
- ANN's moderate performance suggests it captures some non-linear patterns better than basic logistic models, but may struggle with overfitting or optimal hyperparameter tuning compared to the top performers.
- The K-Nearest Neighbors model has the lowest at 0.3697, outperforming all other models including MLP. KNN performs the best is mainly because can capture complex local patterns without making strong assumptions about the overall data distribution.