Predictive Modeling Analysis for “Hot Search”

Joy Wu
May 15th 2024

1. Introduction

In China, "hot search”, or China's News Trending, refers to the most searched keywords or phrases on a website's search engine. These hot searches typically reflect major events or trending topics across various sectors of society during a specific period.
The goal of this project is to utilize machine learning techniques to analyze the characteristics of news related to hot searches published on the Sina Weibo website. The aim is to predict whether a particular news item has the potential to become a hot search topic. This project employed logistic regression, decision trees, random forests, and PCA methods for model optimization, achieving an accuracy of 0.66 and an AUC (Area Under the Curve) value of 0.716 using random forests and Bayesian optimization.

2. Data

• Data Source
The dataset for this project is derived from the hot search data on Sina Weibo from 2021 to 2023. It consists of 59 independent variables representing observed news features and 1 binary target variable indicating whether the news made it to the hot search list. The dataset is a balanced panel with a total of 31,713 observations. It's worth noting that a value of 0 does not signify missing data but rather a numerical value of zero.

• Data Description
Below is the descriptive statistics of the data. The Quantile list represents the minimum, average, and maximum values (the average is used for variables of the same type). It's important to mention the best, worst, and average label counts. These are calculated by sorting all news articles on the website based on the average number of reposts per label.

Index	Label	Description	Quantile
1	time	The older the time from now on, the earlier it will be	8, 339, 731
2	count_head	Number of words in the question	2, 10, 23
3	count_content	Number of words in the main text	0, 409, 7764
4	r_unique	The proportion of non repeating words in the main text	0, 0.54, 0.99
5	r_non_stop	The proportion of non repeating stop words in the main text	0, 0.99, 1
6	r_non_stop_unique	The proportion of non repeating and non stopping words in the main text	0, 0.69, 0.99
7	count_link	The links included in the article	0, 7, 304
8	count_inter_link	Internal links included in the article	0, 2, 74
9	count_images	Number of images included in the article	0, 1, 128
10	count_videos	Number of videos included in the article	0, 0, 91
11	avg_word_len	Average word length	0, 4.66, 7.97
12	count_labels	Number of tags marked by news	1, 7, 10

13-18	forum_i	Does the news belong to the life, entertainment, business, stock, technology, and international sections	0, 0, 1
19-21	worst_label_min/max/avg	Worst tag forwarding times	-1, 236, 39979
22-24	best_label_min/max/avg	Best tag forwarding times	0, 244733, 843300
25-27	arg_label_min/max/avg	Average number of tag forwarding attempts	0, 2872, 36023
28-30	ref_min/max/avg	The number of reposts of internal links in the article	0, 2200, 843300
31-38	wi	Release day of week	0, 0, 1
39-43	dist_i	The probability of news topic being A-E	0.02, 0.04, 0.92
44	total_neutral	News subjectivity	0, 0.45, 1
45	sent_total	News sentiment polarity	-0.39, 0.12, 0.66
46	total_r_pos	The proportion of positive words in the main text	0, 0.04, 0.15
47	total_r_neg	The proportion of negative words in the main text	0, 0.01, 0.02
48	r_pos	The proportion of positive words in non neutral words	0, 0.7, 1
49	r_reg	The proportion of negative words in non neutral words	0, 0.3, 1
50-52	sent_pos_avg/min/max	Positive word emotional polarity	0, 0.35, 1
53-55	sent_neg_avg/min/max	Negative word sentiment polarity	-1, -0.25, 0
56	head_neutral	Title subjectivity	0, 0.15, 1
57	sent_head	Title Emotional Polarity	-1, 0, 1
58	head_neutral_scaled	The absolute difference between the subjectivity of the title and the median (0.5)	0, 0.5, 0.5
59	sent_head_abs	The absolute value of the difference between the emotional polarity of the title and the median value (0)	0, 0, 1
60	hot	Has the news been on hot search	0, 1, 1

• Data Processing

The initial data cleaning process involved several key steps. Firstly, missing values were addressed by identifying and counting the number of missing variables for each observation. Observations with more than 10 missing variables were then excluded from the dataset to ensure data integrity and reliability. Subsequently, variables with all-zero values were removed from the dataset as they did not provide any meaningful information for analysis. Following this, the dataset was divided into two subsets based on data type: one subset contained object type variables while the other contained non-object type variables. Missing values in the non-object type subset were filled using the mean of each column, ensuring that the dataset was complete and ready for further analysis. Finally, the feature matrix was defined by selecting all columns except the target variable, and the target variable itself was defined as the 'hot' column in the dataset. These data cleaning steps were crucial in preparing the dataset for machine learning modeling and analysis, ensuring that the data was accurate, complete, and suitable for predictive modeling tasks.

3. Methods

• Machine Learning

In this project, the overall dataset was divided into training and testing sets, allocating 30% of the data to the validation set for model evaluation while utilizing 70% for model training. Aiming for high predictive accuracy, I used common methods including logistic regression, decision trees, random forests, and PCA for model training.The optimal parameters were obtained using Bayesian optimization, considering the specific characteristics of the data.

• Model Performance Evaluation

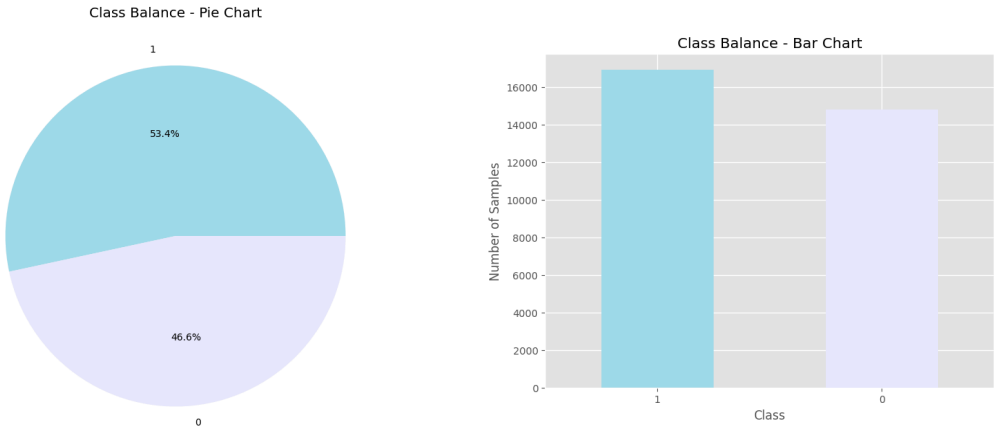
In addition to metrics like accuracy, recall, and F1 score, the Receiver Operating Characteristic (ROC) curve is a valuable tool used to assess the performance of classification models. This curve plots the False Positive Rate (FPR) against the True Positive Rate (TPR) as coordinates, providing a visual representation of how well the model distinguishes between positive and negative samples. The Area Under the ROC Curve (AUC) is then utilized as a metric to quantify the model's performance across various thresholds, indicating its ability to make accurate classifications.

Moreover, Area Under the ROC Curve (AUC) is a critical metric that ranges from 0 to 1, reflecting the performance of classification models. A higher AUC value indicates superior performance in discriminating between positive and negative samples. Specifically, an AUC of 1 signifies a perfect classifier that can differentiate flawlessly, while an AUC above 0.5 indicates performance better than random chance. Conversely, an AUC of 0.5 suggests random guessing, and an AUC below 0.5 implies performance worse than random chance, possibly due to an inverse relationship between data labels and features.

4. Results

• Logistic Regression

First, we visualized the class balance within the target variable of the logistic regression model. The analysis revealed that positive samples (representing trending news) accounted for 53.4% of the dataset, while negative samples (representing non-trending news) made up 46.6%. It's important to note that imbalanced class distributions can impact model training and evaluation. In this dataset, we observed a slight imbalance that may lead the model to be biased towards predicting the dominant class. To address this imbalance and mitigate potential bias, I opted for class weighting, a technique that adjusts the importance of different classes, allowing the model to focus more on minority classes and reduce bias. Additionally, in subsequent models, I incorporated random forests to further tackle the issue of class imbalance.

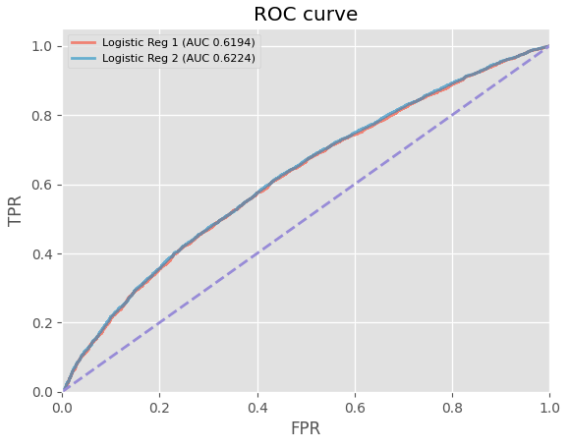


The logistic regression model was trained and evaluated using default parameters as well as optimized parameters. The default parameters yielded an accuracy of approximately 59% on the training data and 58% on the validation data. After parameter tuning, the optimized logistic regression model achieved similar performance metrics.

On the training set, the model resulted in a precision of 0.58, recall of 0.79, and an F1-score of 0.67 for class 1 (news trending on Sina Weibo). The overall accuracy was 0.59. On the validation set, the precision, recall, and F1-score for class 1 were 0.58, 0.79, and 0.67, respectively. The model's accuracy on the validation set was 0.58.

The Receiver Operating Characteristic (ROC) curve plotted for both the default and optimized logistic regression models exhibited similar curves, with the AUC values aligning closely around 0.62. This indicates a moderate discrimination ability of the models in distinguishing between news articles that trended on Sina Weibo and those that did not.

Overall, the logistic regression models showed moderate predictive performance, with room for improvement in terms of accuracy and discrimination ability. Further model refinement and feature engineering may enhance the model's predictive capabilities.



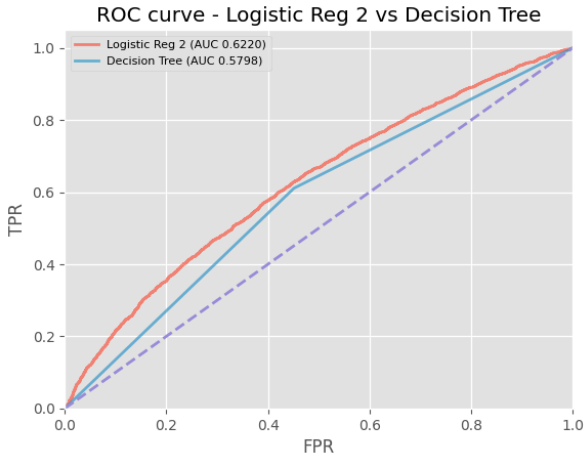
• Decision Tree

The decision tree model achieved high accuracy and performance metrics in both training and validation sets. In the training set, the model achieved perfect accuracy (1.00) and F1-score for both positive and negative samples, indicating excellent classification ability. The precision, recall, and F1-score for the positive class were 1.00, demonstrating the model's ability to correctly identify trending news articles.

In the validation set, the decision tree model demonstrated relatively good performance with an accuracy of 0.59 and F1-score of 0.61 for the positive class. The precision and recall for the positive class were also reasonably balanced at 0.61.

The AUC value for the decision tree model was approximately 0.58, indicating a moderate level of discriminative ability.

Comparing the ROC curves of Logistic Regression with optimal parameters and the decision tree model, both models demonstrated similar AUC values, suggesting comparable performance in terms of classification ability. However, the decision tree model exhibited a different balance between precision, recall, and F1-score, showcasing its unique strengths and weaknesses in handling the dataset.

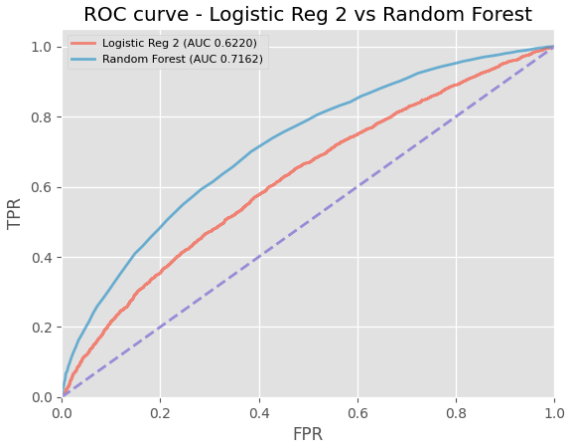


• Random forest

The random forest model was optimized using the Bayesian method to achieve optimal results for this project. The optimization process focused on maximizing the F1 score, weighted across classes, on the validation set. The best parameters, including class weights of {0: 0.45892, 1: 0.55449} and a random state of 480, were determined through Bayesian optimization and directly applied to the model to enhance its performance.

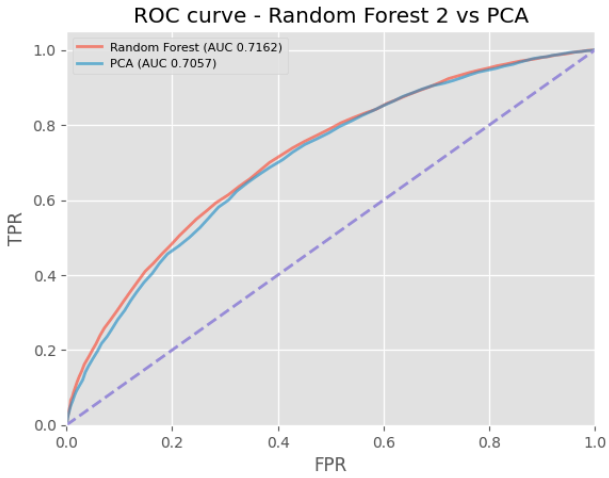
The optimized Random Forest model demonstrated outstanding performance on both the training and validation sets. In the training set, the model achieved perfect accuracy (1.00) and F1-scores for both positive and negative samples, highlighting its robust classification capability. Specifically, the precision, recall, and F1-score for the positive class were all 1.00, indicating precise identification of positive instances without missing any.

Moving to the validation set, the optimized Random Forest model maintained strong performance with an accuracy of 0.66 and an F1-score of 0.69 for the positive class. These metrics suggest that the model generalized well to unseen data while retaining high predictive accuracy. The balance between precision (0.68) and recall (0.68) for the positive class further underscores the model's ability to correctly identify true positives while minimizing false positives.
Moreover, the AUC value for the optimized Random Forest model was approximately 0.78, signifying a good level of discriminative ability in distinguishing between positive and negative samples. This improvement in AUC compared to other models reflects the effectiveness of the optimization process in enhancing the model's overall performance on the given dataset.



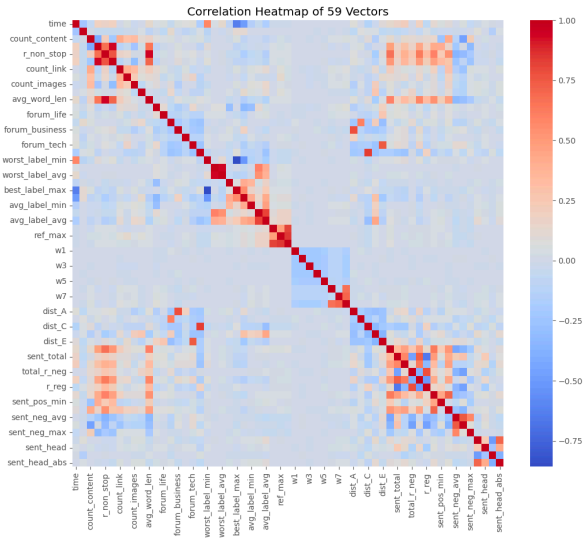
• Random Forest with PC

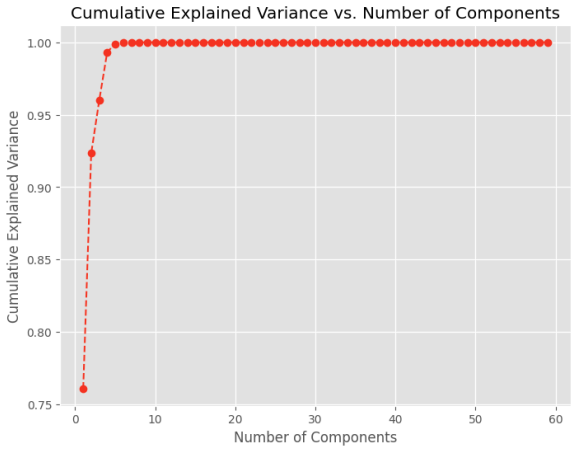
The RF-PCA model achieved satisfactory results in both the training and validation sets. In validation set, the RL-PCA model demonstrated an accuracy of 0.66 and an F1-score of 0.65 for the positive class. These metrics suggest that the RF-PCA model generalized reasonably well to unseen data while maintaining a relatively high level of predictive accuracy. The precision and recall for the positive class were also balanced at 0.67.
Comparing the RF-PCA model with the RF model, it is observed that the RF model achieved better performance across multiple metrics, including precision, recall, and F1-score for the positive class. This difference in performance may be attributed to the reduction in feature informativeness caused by PCA, leading to a slight decrease in model performance.



Current model utilized 25 principal components derived from the original 59 variables in the dataset. Notably, including up to the fifth principal component was sufficient to capture over 99% of the cumulative explained variance, demonstrating effective dimensionality reduction while retaining the majority of the dataset's information.

Moreover, the correlation analysis among variables in the dataset revealed low correlations between variables, indicating a lack of strong linear relationships among features. This low correlation may have contributed to the decrease in performance observed in the RL-PCA model, as PCA relies on capturing linear relationships between variables for effective dimensionality reduction. However, despite the reduction in performance compared to the RL model, the RL-PCA model still achieved reasonable accuracy and F1-score, suggesting its potential utility in scenarios where dimensionality reduction is critical despite a slight trade-off in predictive performance.





5. Conclusion

The optimized Random Forest model emerged as the top performer in this predictive modeling analysis, showcasing superior predictive accuracy and generalization to unseen data. With an accuracy of 0.66 and an F1-score of 0.69 for the positive class, the model demonstrated robust classification ability. Moreover, the precision and recall for the positive class were balanced at 0.68, indicating the model's capability to correctly identify true positives while minimizing false positives. The AUC value of approximately 0.78 further underscored the model's discriminative ability in distinguishing between positive and negative samples.

Further exploration and refinement of models, along with feature engineering and dimensionality reduction techniques, can enhance predictive capabilities and provide more accurate predictions for news articles' trending potential on Sina Weibo. Specifically, addressing the imbalanced class distribution, low correlation among variables, and incorporating relevant features such as repost counts, sentiment analysis, and topic categorization can improve model performance and accuracy. Additionally, fine-tuning parameters, optimizing class weights, and exploring advanced machine learning algorithms tailored to the dataset's characteristics can contribute to better predictive outcomes and insights into trending topics on the platform.

Appendix

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use("ggplot")

# getdata
riskdat =  pd.read_csv('数据集.csv')
riskdat_c = riskdat.copy()

# dim: 31713 samples, 59 features, 1 label
print(riskdat.shape)
print(riskdat.head())
# + - distribution
print(riskdat.hot.value_counts())
# missing variables
riskdat.missing_var = riskdat.isnull().sum(axis=1)
riskdat=riskdat.loc[riskdat.missing_var < 10,:]
print(riskdat.shape)
# delete all-zero variables
riskdat=riskdat.loc[:,~(riskdat == 0).all(axis=0)]
riskdat.shape
# delete non-data variables
riskdat_sub2 = riskdat.select_dtypes(include=['object'])
riskdat_sub2.head(3)
riskdat_sub1=riskdat.select_dtypes(exclude=['object'])
riskdat_sub1.head(3)
riskdat_sub1 = riskdat_sub1.fillna(riskdat_sub1.mean())
print(riskdat_sub1.shape)

# define X and y
X1, y1 = riskdat_sub1.iloc[:, 0:-1], riskdat_sub1.hot
print(X1.shape, y1.shape)

# import libraries for modeling
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from bayes_opt import BayesianOptimization
from sklearn.metrics import f1_score
from sklearn.decomposition import PCA
import seaborn as sns

# slpit training set and test set
X1_train, X1_val, y1_train, y1_val = train_test_split(X1.values, y1.values, test_size=0.3, random_state=114)
print(X1_train.shape, y1_train.shape, X1_val.shape, y1_val.shape)

# data discription
DD = riskdat_sub1.describe()
DD
```

```
# %%
# class balance (positive and negetive) _for 1^ LOGISTIC REGRESSION
class_balance = y1.value_counts()

# Bar chart by counts
plt.figure(figsize=(8, 6))
class_balance.plot(kind='bar', color=['lightblue', 'lavender'])
plt.xlabel('Class')
plt.ylabel('Number of Samples')
plt.title('Class Balance - Bar Chart')
plt.xticks(rotation=0)
plt.show()

# Pie chart by proportion
plt.figure(figsize=(8, 8))
class_balance.plot(kind='pie', autopct='%1.1f%%', colors=['lightblue', 'lavender'])
plt.title('Class Balance - Pie Chart')
plt.ylabel('')
plt.show()

# %%
# 1^ LOGISTIC REGRESSION

# 1.1^ default parms
lr_clf1 = LogisticRegression(class_weight={0: 0.47, 1: 0.53})
lr_clf1.fit(X1_train, y1_train)
y1_train_pred = lr_clf1.predict(X1_train)
print("Confusion matrix (training):\n {0}\n".format(confusion_matrix(y1_train, y1_train_pred)))
print("Classification report (training):\n {0}".format(classification_report(y1_train, y1_train_pred)))

y1_val_pred = lr_clf1.predict(X1_val)
print("Confusion matrix (validation):\n {0}\n".format(confusion_matrix(y1_val, y1_val_pred)))
print("Classification report (validation):\n {0}".format(classification_report(y1_val, y1_val_pred)))

# prams tuning
lr_clf_tuned = LogisticRegression(class_weight={0: 0.47, 1: 0.53})
lr_clf_params = {
   "penalty": ["l1", "l2"],
    "C": [1, 1.3, 1.5, 2]
}
lr_clf_cv = GridSearchCV(lr_clf_tuned, lr_clf_params, cv=5)
lr_clf_cv.fit(X1_train, y1_train)
print(lr_clf_cv.best_params_)

# 1.2^ optimal parms
lr_clf2 = LogisticRegression(penalty="l2", C=2, class_weight={0: 0.47, 1: 0.53})
lr_clf2.fit(X1_train, y1_train)
y1_train_pred = lr_clf2.predict(X1_train)
print("Confusion matrix (training):\n {0}\n".format(confusion_matrix(y1_train, y1_train_pred)))
print("Classification report (training):\n {0}".format(classification_report(y1_train, y1_train_pred)))

y1_val_pred = lr_clf2.predict(X1_val)
print("Confusion matrix (validation):\n {0}\n".format(confusion_matrix(y1_val, y1_val_pred)))
print("Classification report (validation):\n {0}".format(classification_report(y1_val, y1_val_pred)))

# ROC
```

```
y1_valid_score_lr1 = lr_clf1.predict_proba(X1_val)
y1_valid_score_lr2 = lr_clf2.predict_proba(X1_val)

fpr_lr1, tpr_lr1, thresholds_lr1 = roc_curve(y1_val, y1_valid_score_lr1[:, 1])
fpr_lr2, tpr_lr2, thresholds_lr2 = roc_curve(y1_val, y1_valid_score_lr2[:, 1])

roc_auc_lr1 = auc(fpr_lr1, tpr_lr1)
roc_auc_lr2 = auc(fpr_lr2, tpr_lr2)

plt.plot(fpr_lr1, tpr_lr1, fpr_lr2, tpr_lr2, lw=2, alpha=.6)
plt.plot([0, 1], [0, 1], lw=2, linestyle="--")
plt.xlim([0, 1])
plt.ylim([0, 1.05])
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve")
plt.legend(["Logistic Reg 1 (AUC {:.4f})".format(roc_auc_lr1),
            "Logistic Reg 2 (AUC {:.4f})".format(roc_auc_lr2)], fontsize=8, loc=2)

plt.show()
```

```
# %%
# 2^ Disision Tree
dt_clf = DecisionTreeClassifier(class_weight={0: 0.47, 1: 0.53}, random_state=123)
dt_clf.fit(X1_train, y1_train)

y1_train_pred_dt = dt_clf.predict(X1_train)
print("Confusion matrix (training - Decision Tree):\n", confusion_matrix(y1_train, y1_train_pred_dt))
print("Classification report (training - Decision Tree):\n", classification_report(y1_train, y1_train_pred_dt))

y1_val_pred_dt = dt_clf.predict(X1_val)
print("Confusion matrix (validation - Decision Tree):\n", confusion_matrix(y1_val, y1_val_pred_dt))
print("Classification report (validation - Decision Tree):\n", classification_report(y1_val, y1_val_pred_dt))

y1_valid_score_dt = dt_clf.predict_proba(X1_val)

fpr_dt, tpr_dt, thresholds_dt = roc_curve(y1_val, y1_valid_score_dt[:, 1])
roc_auc_dt = auc(fpr_dt, tpr_dt)

plt.plot(fpr_lr2, tpr_lr2, fpr_dt, tpr_dt, lw=2, alpha=.6)
plt.plot([0, 1], [0, 1], lw=2, linestyle="--")
plt.xlim([0, 1])
plt.ylim([0, 1.05])
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve - Logistic Reg 2 vs Decision Tree")
plt.legend(["Logistic Reg 2 (AUC {:.4f})".format(roc_auc_lr2),
            "Decision Tree (AUC {:.4f})".format(roc_auc_dt)], fontsize=8, loc=2)

plt.show()
```

```
# %%
# parms tunning (Bayesian) _ for 3^ RANDOM FOREST
def optimize_rf(class_weight_0, class_weight_1, random_state):
    class_weight = {0: class_weight_0, 1: class_weight_1}
    rf_clf = RandomForestClassifier(class_weight=class_weight, random_state=int(random_state))
    rf_clf.fit(X1_train, y1_train)
    y_val_pred = rf_clf.predict(X1_val)
    score = -f1_score(y1_val, y_val_pred, average='weighted')
    return score

pbounds = {'class_weight_0': (0.4, 0.6),
           'class_weight_1': (0.4, 0.6),
           'random_state': (0, 1000)}
optimizer = BayesianOptimization(
    f=optimize_rf,
    pbounds=pbounds,
    random_state=123,
)
optimizer.maximize(init_points=5, n_iter=10)
best_params = optimizer.max['params']
print(best_params)
```

```
# %%
# 3^ RANDOM FOREST
rf_clf = RandomForestClassifier(class_weight={0: 0.45892, 1: 0.55449}, random_state=480)
rf_clf.fit(X1_train, y1_train)

y1_train_pred_rf = rf_clf.predict(X1_train)
print("Confusion matrix (training - Random Forest):\n", confusion_matrix(y1_train, y1_train_pred_rf))
print("Classification report (training - Random Forest):\n", classification_report(y1_train, y1_train_pred_rf))

y1_val_pred_rf = rf_clf.predict(X1_val)
print("Confusion matrix (validation - Random Forest):\n", confusion_matrix(y1_val, y1_val_pred_rf))
print("Classification report (validation - Random Forest):\n", classification_report(y1_val, y1_val_pred_rf))

y1_valid_score_rf = rf_clf.predict_proba(X1_val)

fpr_rf, tpr_rf, thresholds_rf = roc_curve(y1_val, y1_valid_score_rf[:, 1])
roc_auc_rf = auc(fpr_rf, tpr_rf)

plt.plot(fpr_lr2, tpr_lr2, fpr_rf, tpr_rf, lw=2, alpha=.6)
plt.plot([0, 1], [0, 1], lw=2, linestyle="--")
plt.xlim([0, 1])
plt.ylim([0, 1.05])
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve - Logistic Reg 2 vs Random Forest")
plt.legend(["Logistic Reg 2 (AUC {:.4f})".format(roc_auc_lr2),
            "Random Forest (AUC {:.4f})".format(roc_auc_rf)], fontsize=8, loc=2)

plt.show()
```

```
# %%
# 4^ RANDOM FOREST wiht PCA
pca = PCA(n_components=25)
X1_train_pca = pca.fit_transform(X1_train)
X1_val_pca = pca.transform(X1_val)

rf_clf_pca = RandomForestClassifier(class_weight={0: 0.45892, 1: 0.55449}, random_state=480)
rf_clf_pca.fit(X1_train_pca, y1_train)

y1_train_pred_pca = rf_clf_pca.predict(X1_train_pca)
print("Confusion matrix (training - Random Forest):\n", confusion_matrix(y1_train, y1_train_pred_pca))
print("Classification report (training - Random Forest):\n", classification_report(y1_train, y1_train_pred_pca))

y1_val_pred_pca = rf_clf_pca.predict(X1_val_pca)
print("Confusion matrix (validation - Random Forest):\n", confusion_matrix(y1_val, y1_val_pred_pca))
print("Classification report (validation - Random Forest):\n", classification_report(y1_val, y1_val_pred_pca))

y1_valid_score_pca = rf_clf_pca.predict_proba(X1_val_pca)

fpr_pca, tpr_pca, thresholds_pca = roc_curve(y1_val, y1_valid_score_pca[:, 1])
roc_auc_pca = auc(fpr_pca, tpr_pca)

plt.plot(fpr_rf, tpr_rf, fpr_pca, tpr_pca, lw=2, alpha=.6)
plt.plot([0, 1], [0, 1], lw=2, linestyle="--")
plt.xlim([0, 1])
plt.ylim([0, 1.05])
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve - Random Forest 2 vs PCA")
plt.legend(["Random Forest (AUC {:.4f})".format(roc_auc_rf),
            "PCA (AUC {:.4f})".format(roc_auc_pca)], fontsize=8, loc=2)

plt.show()


# %%
# choose features _for 4^ RANDOM FOREST with PCA
pca = PCA()
pca.fit(X1_train)

# accum explained variance curve
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1),
         np.cumsum(pca.explained_variance_ratio_),
         marker='o', linestyle='--')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Cumulative Explained Variance vs. Number of Components')
plt.grid(True)
plt.show()

# correlation heatmap
corr_matrix = X1.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of 59 Vectors')
plt.show()
```