Description of Implementation

Group 21

We use Python to implement this project. The packages used are mainly pandas, numpy and sklearn. We use pandas and numpy to show the features of variables, use sklearn to build models, and also use matplotlib for visualization.

Evaluation Method

We use AUCROC to evaluate the performance of models by a function **aucroc_score(model, cv)**, which runs **sklearn.model_selection.cross_val_score(scoring=' roc_auc')**.

Models Implementation

1) Linear Regression

We use regularization to tackle the overfitting problem, including Ridge and Lasso Regression.

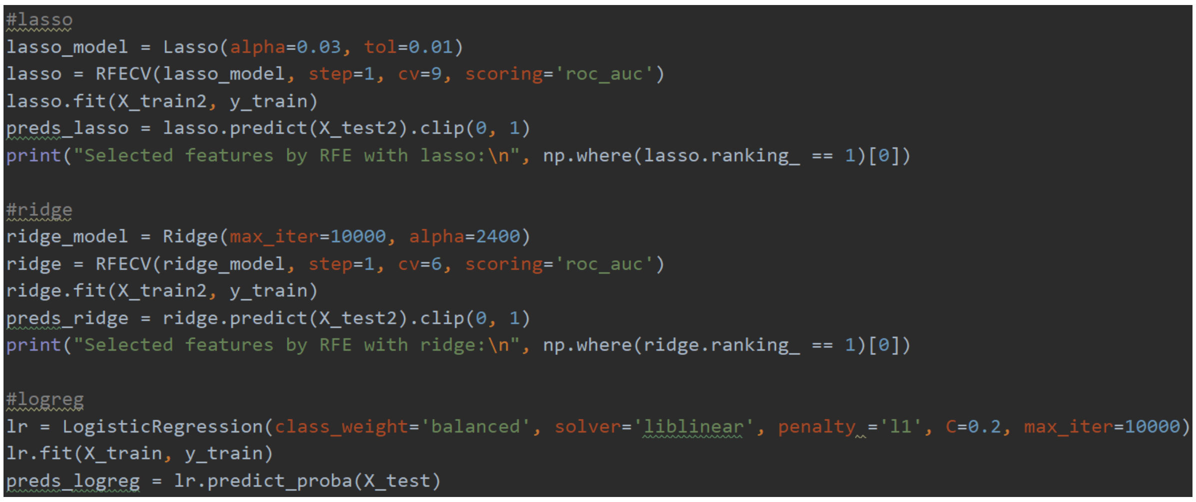
Using **RidgeCV** and **LassoCV** in **sklearn.linear_model** with a list of parameters λ, we can find out the best λ, which is 0.03. Besides, we also use a for-loop to find out the best value of cross-validation (**cv**), which is 9 for Lasso and 6 for Ridge. With determined parameters, we build the model and use **RFECV** to select features. After fitting the models, we can generate the prediction results.

2) Logistic Regression

We also try logistic regression, because it models the probability and the target is binary in this case.

Similarly, using **LogisticRegressionCV** with parameter Cs, a list of C describing the inverse of regularization strength, we get the best parameter C=0.2. And L1 (Lasso) penalty is selected here for better performance. Then we simply use the features obtained before to fit the model.

The final model implementation code is as follows.



```
#lasso
lasso_model = Lasso(alpha=0.03, tol=0.01)
lasso = RFECV(lasso_model, step=1, cv=9, scoring='roc_auc')
lasso.fit(X_train2, y_train)
preds_lasso = lasso.predict(X_test2).clip(0, 1)
print("Selected features by RFE with lasso:\n", np.where(lasso.ranking_ == 1)[0])

#ridge
ridge_model = Ridge(max_iter=10000, alpha=2400)
ridge = RFECV(ridge_model, step=1, cv=6, scoring='roc_auc')
ridge.fit(X_train2, y_train)
preds_ridge = ridge.predict(X_test2).clip(0, 1)
print("Selected features by RFE with ridge:\n", np.where(ridge.ranking_ == 1)[0])

#logreg
lr = LogisticRegression(class_weight='balanced', solver='liblinear', penalty ='l1', C=0.2, max_iter=10000)
lr.fit(X_train, y_train)
preds_logreg = lr.predict_proba(X_test)
```

Results

We obtain scores (private, public): (0.852, 0.862), (0.849, 0.859) and (0.838, 0.850) for logistic, lasso and ridge regression respectively. Considering the close scores, we combine them together with different weights: [0.4, 0.2, 0.4], and get a higher score: (0.856, 0.868).