Machine Learning Final Project

109550187 梁巍濤

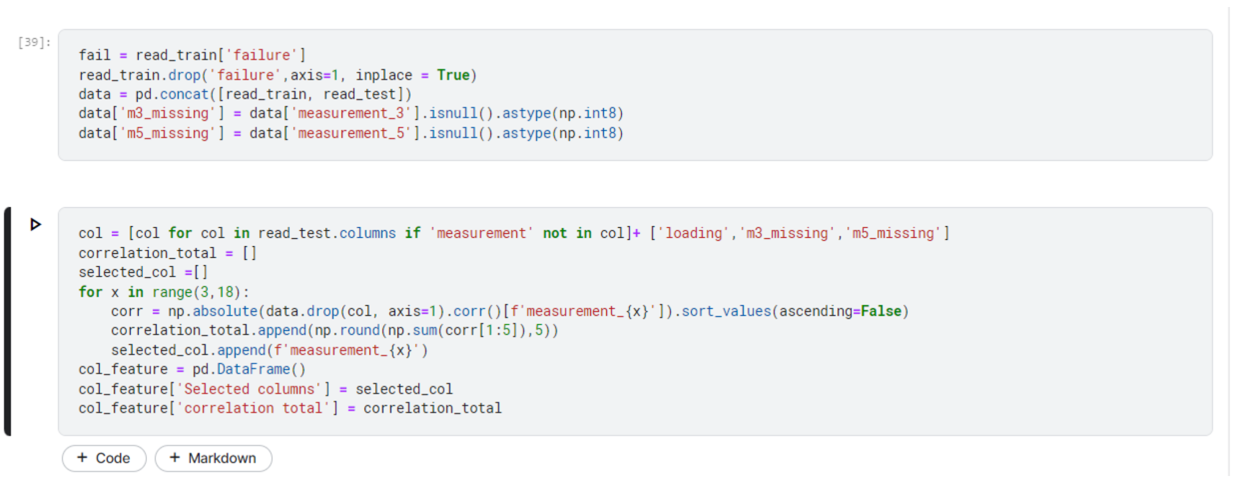
Github link of project:
https://github.com/109550187/MACHINELEARNINGFINAL

The final project asks us to join a real life competition to predict a failure rate for each id in the test set. The data that we are given of are three csv files in the form of train and test csv's, and also the sample submission csv file as the format template for the submission.

For the first couple of days I tried to research what exactly and how we are supposed to do the competition itself. I ended up working on my code based on a few reference codes that I've seen in the Kaggle competition itself. https://www.kaggle.com/code/saraswatitiwari/tabular-playground-series-august-2022 is the code that I've referenced and worked with using a different approach.

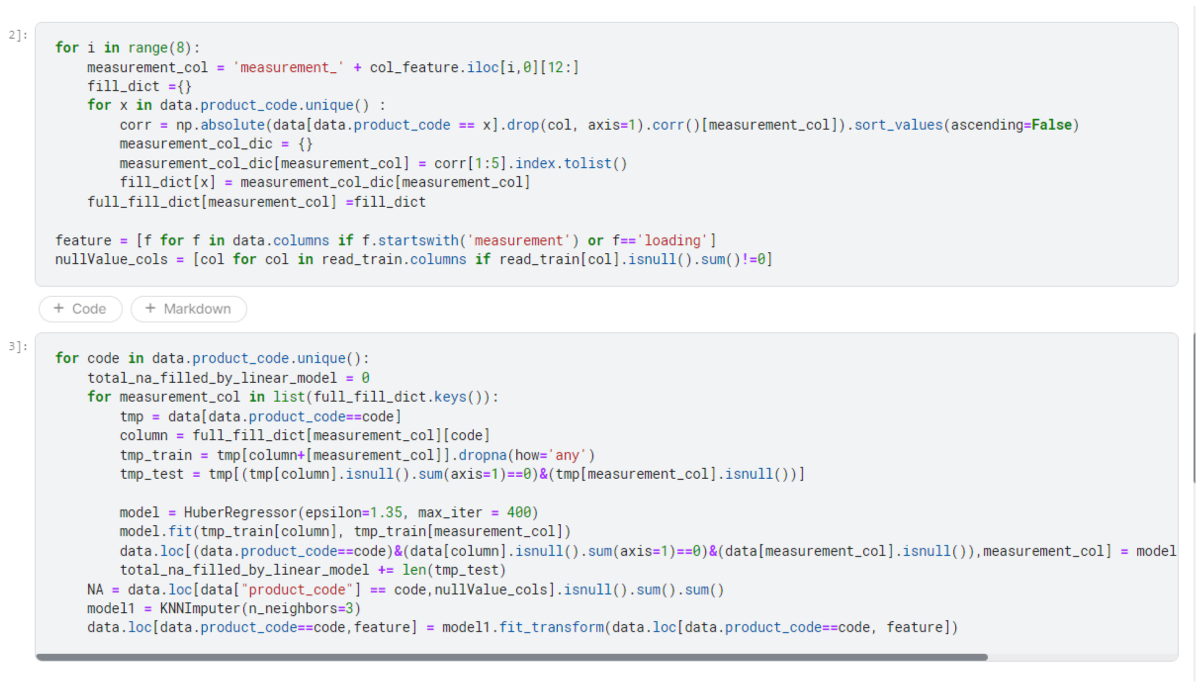
I worked the project using the test and train csv's provided by the Kaggle competition itself, and edited the code inside of Kaggle's code interface. I used Logistic Regression as an approach to solve this problem. Logistic regression is a statistical technique used to analyze the relationship between a given set of independent variables and a binary dependent variable. It is widely used to model the probability of an event occurring or not occurring. In logistic regression, a logit transformation is applied on the odds, which is the probability of success divided by the probability of failure. This transformation creates a linear relationship between the independent variables and the log odds, which is then used to calculate probabilities.

Logistic regression is a type of supervised machine learning algorithm and can be used to model the probability of a certain class or event. It is used to calculate or predict the probability of a binary (yes/no) event occurring. For example, it can be used to predict whether an email is spam or not spam, or if a tumor is malignant or not malignant. It is also used to interpret the strength of influence a person's age, gender, and dating status may have on the type of film they prefer.



```
fail = read_train['failure']
read_train.drop('failure',axis=1, inplace = True)
data = pd.concat([read_train, read_test])
data['m3_missing'] = data['measurement_3'].isnull().astype(np.int8)
data['m5_missing'] = data['measurement_5'].isnull().astype(np.int8)
```

```
col = [col for col in read_test.columns if 'measurement' not in col]+ ['loading','m3_missing','m5_missing']
correlation_total = []
selected_col =[]
for x in range(3,18):
    corr = np.absolute(data.drop(col, axis=1).corr()[f'measurement_{x}']).sort_values(ascending=False)
    correlation_total.append(np.round(np.sum(corr[1:5]),5))
    selected_col.append(f'measurement_{x}')
col_feature = pd.DataFrame()
col_feature['Selected columns'] = selected_col
col_feature['correlation total'] = correlation_total
```



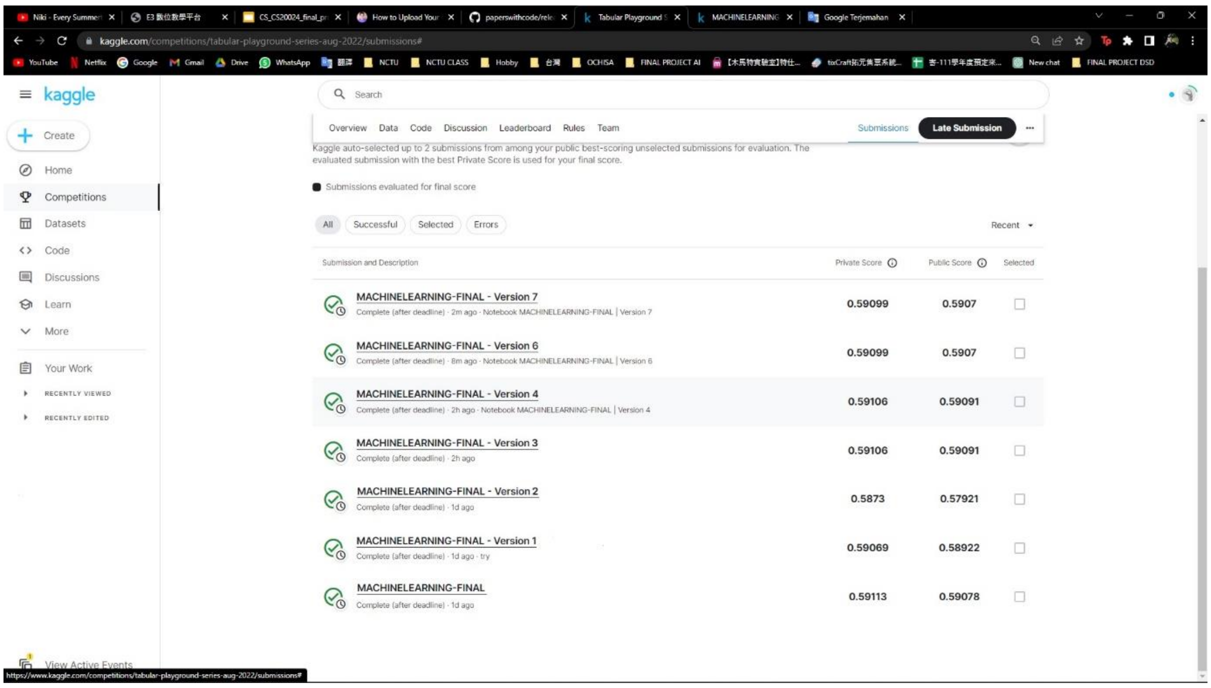
```
2]:
for i in range(8):
    measurement_col = 'measurement_' + col_feature.iloc[i,0][12:]
    fill_dict ={}
    for x in data.product_code.unique() :
        corr = np.absolute(data[data.product_code == x].drop(col, axis=1).corr()[measurement_col]).sort_values(ascending=False)
        measurement_col_dic = {}
        measurement_col_dic[measurement_col] = corr[1:5].index.tolist()
        fill_dict[x] = measurement_col_dic[measurement_col]
    full_fill_dict[measurement_col] =fill_dict

feature = [f for f in data.columns if f.startswith('measurement') or f=='loading']
nullValue_cols = [col for col in read_train.columns if read_train[col].isnull().sum()!=0]
```

```
3]:
for code in data.product_code.unique():
    total_na_filled_by_linear_model = 0
    for measurement_col in list(full_fill_dict.keys()):
        tmp = data[data.product_code==code]
        column = full_fill_dict[measurement_col][code]
        tmp_train = tmp[column+[measurement_col]].dropna(how='any')
        tmp_test = tmp[(tmp[column].isnull().sum(axis=1)==0)&(tmp[measurement_col].isnull())]

        model = HuberRegressor(epsilon=1.35, max_iter = 400)
        model.fit(tmp_train[column], tmp_train[measurement_col])
        data.loc[(data.product_code==code)&(data[column].isnull().sum(axis=1)==0)&(data[measurement_col].isnull()),measurement_col] = model
        total_na_filled_by_linear_model += len(tmp_test)
    NA = data.loc[data["product_code"] == code,nullValue_cols].isnull().sum().sum()
    model1 = KNNImputer(n_neighbors=3)
    data.loc[data.product_code==code,feature] = model1.fit_transform(data.loc[data.product_code==code, feature])
```

Some snippets of my code can be seen in the above screenshots.



kaggle.com/competitions/tabular-playground-series-aug-2022/submissions#

Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

● Submissions evaluated for final score

Submission and Description	Private Score	Public Score	Selected
MACHINELEARNING-FINAL - Version 7 Complete (after deadline) · 2m ago · Notebook MACHINELEARNING-FINAL | Version 7	0.59099	0.5907	☐
MACHINELEARNING-FINAL - Version 6 Complete (after deadline) · 8m ago · Notebook MACHINELEARNING-FINAL | Version 6	0.59099	0.5907	☐
MACHINELEARNING-FINAL - Version 4 Complete (after deadline) · 2h ago · Notebook MACHINELEARNING-FINAL | Version 4	0.59106	0.59091	☐
MACHINELEARNING-FINAL - Version 3 Complete (after deadline) · 2h ago	0.59106	0.59091	☐
MACHINELEARNING-FINAL - Version 2 Complete (after deadline) · 1d ago	0.5873	0.57921	☐
MACHINELEARNING-FINAL - Version 1 Complete (after deadline) · 1d ago · try	0.59069	0.58922	☐
MACHINELEARNING-FINAL Complete (after deadline) · 1d ago	0.59113	0.59078	☐

The scores of my submissions can be seen in the screenshot above. I tried implementing different approaches to get high scores but I ended up achieving only just a slightly above baseline score.

Since I’ve only used test and train csv’s from the original Kaggle post and nothing else need to be downloaded for my code to run, I have downloaded the test, train, and sample submission csv files and you can find them here https://drive.google.com/file/d/1eeJveV3LumJorzCRtSRiMxBOPZIP9ZtY/view?usp=sharing

This should be enough for my code to run in another system.