Round II (Junior ML Engineer)

Shohin Sobirov

DATA COLLECTION

- Data was collected from “https://03online.com”
- 5 classification groups were selected and numbers from 0-4 were assigned to each group(**Pulmonologist(0), Allergist(1), Cardiologist(2), Covid-19 Specialist(3), Dermatologist(4)**)
- Two parameters were collected: Patients’ questions(**X**), predefined group(**Y**).
- At first **500** samples were collected for each group. However, the number of samples were increased to **2500** for each group. Totally around **12500** samples were collected.

Data Preparation

Patients' questions were normalized using these operations:

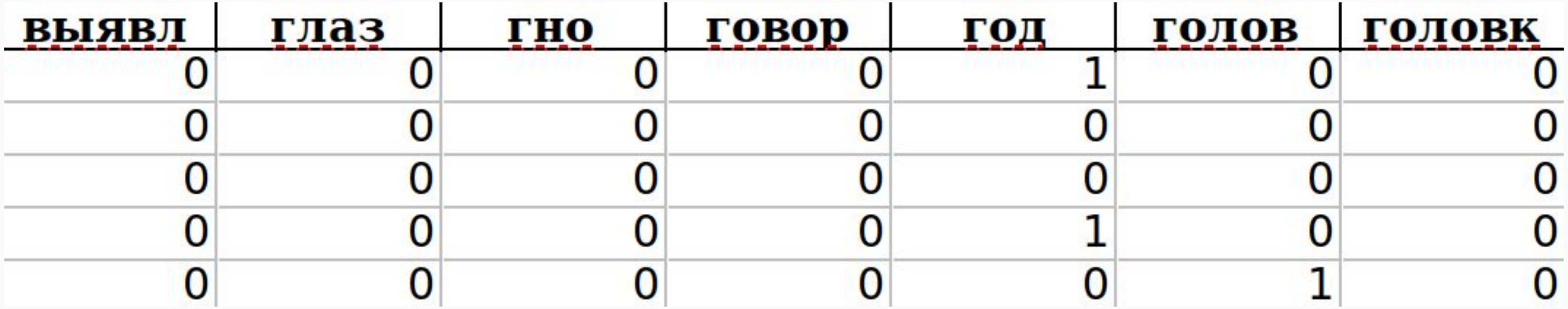
1. All sentences were lower-cased
2. Punctuation signs were removed.
3. Stop_words were removed.
4. Words were stemmed.

Data Preparation

Bag of Words

1. The frequency of each word was calculated.
2. According to these frequencies, most frequent words were selected for each group.
3. CountVectorizer() was used to create Bag of Words from most frequent words
4. All patients’ questions were vectorized to 0’s and 1’s

Data Preparation



выявл	глаз	гно	говор	год	голов	головк
0	0	0	0	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0

CHOOSING A MODEL

These classification models were selected:

1. Naive Bayes
2. RandomForest Classifier
3. DecisionTree Classifier
4. Logistic Regression
5. RandomForest XGBoost
6. GradientBoosting Classifier
7. HistGradientBoosting Classifier
8. CatBoost Classifier
9. LightGBM Classifier

Training

Dataset was divided into Training set and Testing set with 80% for Training and 20% for Testing

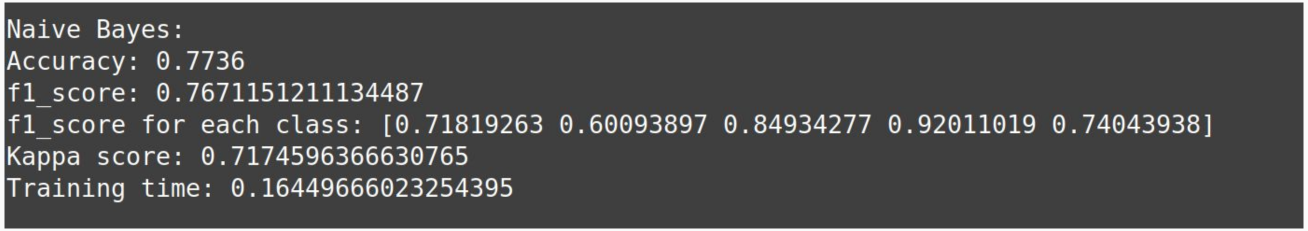
Evaluation

The metrics that were used to compare the performance of each model:

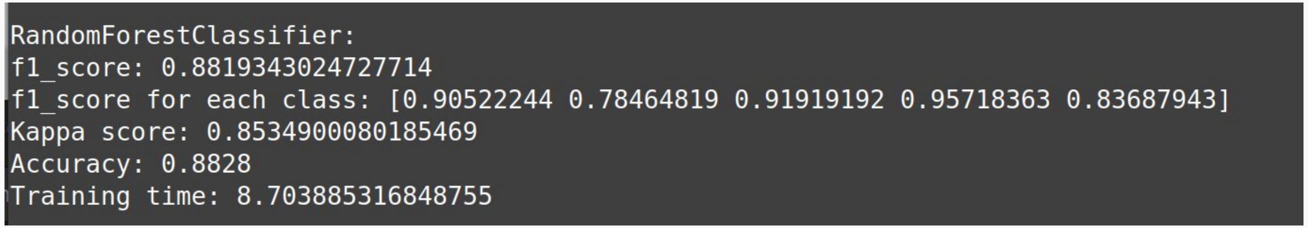
- F1_score(for each group)
- F1_score(weighted)
- Kappa score
- Accuracy
- Training Time

Evaluation

Naive Bayes and RandomForestClassifier



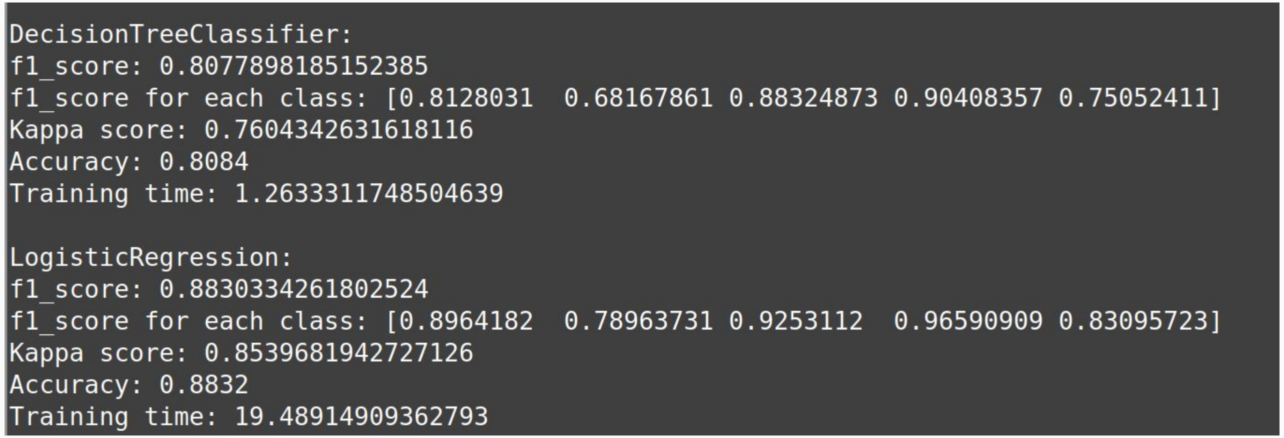
```
Naive Bayes:
Accuracy: 0.7736
f1_score: 0.7671151211134487
f1_score for each class: [0.71819263 0.60093897 0.84934277 0.92011019 0.74043938]
Kappa score: 0.7174596366630765
Training time: 0.16449666023254395
```



```
RandomForestClassifier:
f1_score: 0.8819343024727714
f1_score for each class: [0.90522244 0.78464819 0.91919192 0.95718363 0.83687943]
Kappa score: 0.8534900080185469
Accuracy: 0.8828
Training time: 8.703885316848755
```

Evaluation

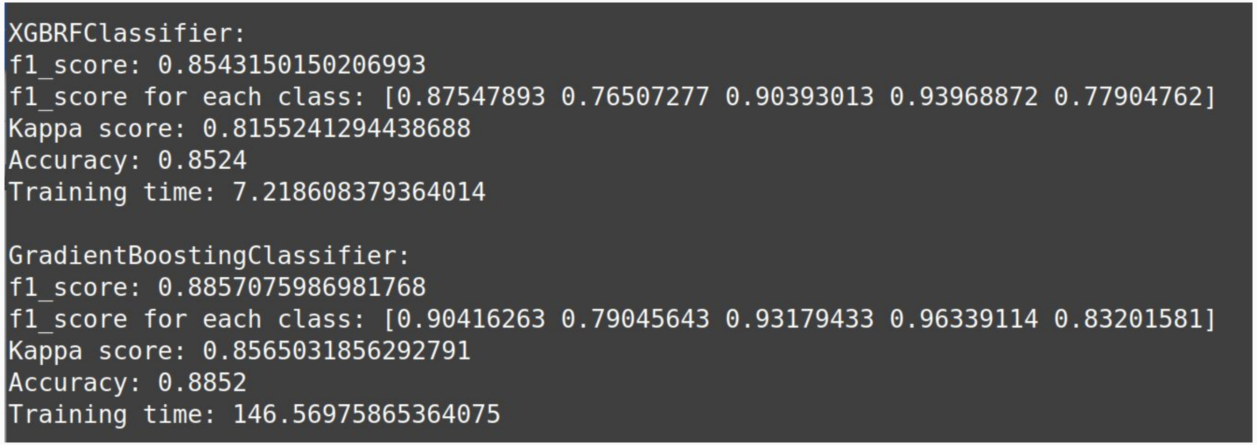
DecisionTreeClassifier and LogisticRegression



```
DecisionTreeClassifier:
f1_score: 0.8077898185152385
f1_score for each class: [0.8128031  0.68167861 0.88324873 0.90408357 0.75052411]
Kappa score: 0.7604342631618116
Accuracy: 0.8084
Training time: 1.2633311748504639

LogisticRegression:
f1_score: 0.8830334261802524
f1_score for each class: [0.8964182  0.78963731 0.9253112  0.96590909 0.83095723]
Kappa score: 0.8539681942727126
Accuracy: 0.8832
Training time: 19.48914909362793
```

Evaluation

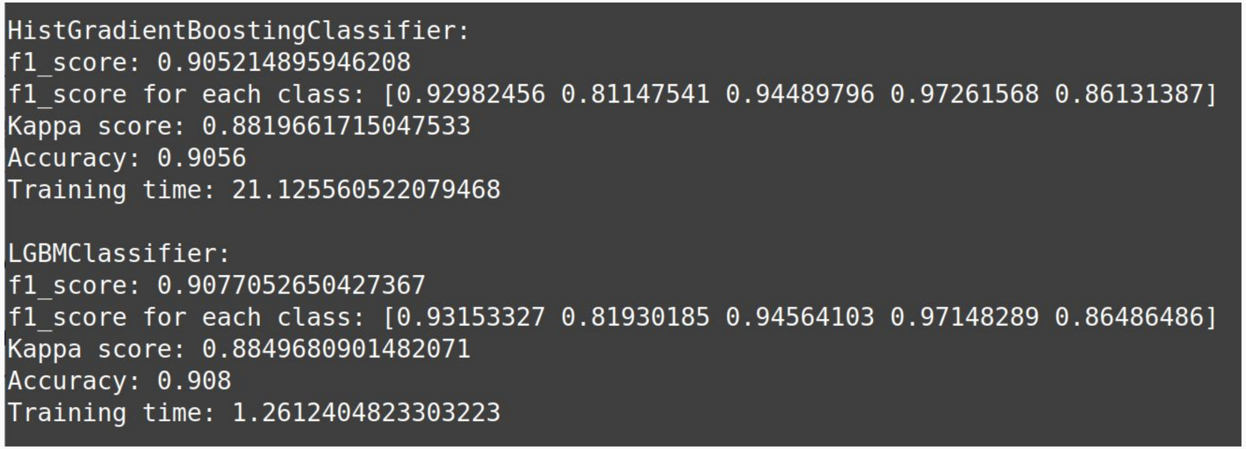
RandomForest XGBoost and GradientBoostingClassifier



```
XGBRFClassifier:
f1_score: 0.8543150150206993
f1_score for each class: [0.87547893 0.76507277 0.90393013 0.93968872 0.77904762]
Kappa score: 0.8155241294438688
Accuracy: 0.8524
Training time: 7.218608379364014

GradientBoostingClassifier:
f1_score: 0.8857075986981768
f1_score for each class: [0.90416263 0.79045643 0.93179433 0.96339114 0.83201581]
Kappa score: 0.8565031856292791
Accuracy: 0.8852
Training time: 146.56975865364075
```

Evaluation

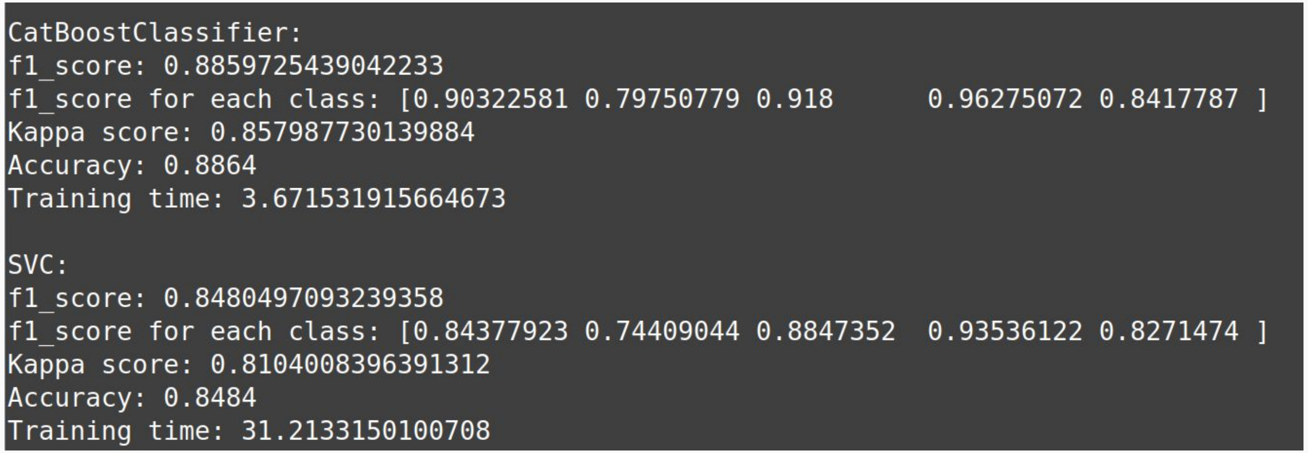
HistGradientBoostingClassifier and LightGBMClassifier



```
HistGradientBoostingClassifier:
f1_score: 0.905214895946208
f1_score for each class: [0.92982456 0.81147541 0.94489796 0.97261568 0.86131387]
Kappa score: 0.8819661715047533
Accuracy: 0.9056
Training time: 21.125560522079468

LGBMClassifier:
f1_score: 0.9077052650427367
f1_score for each class: [0.93153327 0.81930185 0.94564103 0.97148289 0.86486486]
Kappa score: 0.8849680901482071
Accuracy: 0.908
Training time: 1.2612404823303223
```

Evaluation

CatBoostClassifier and SVC



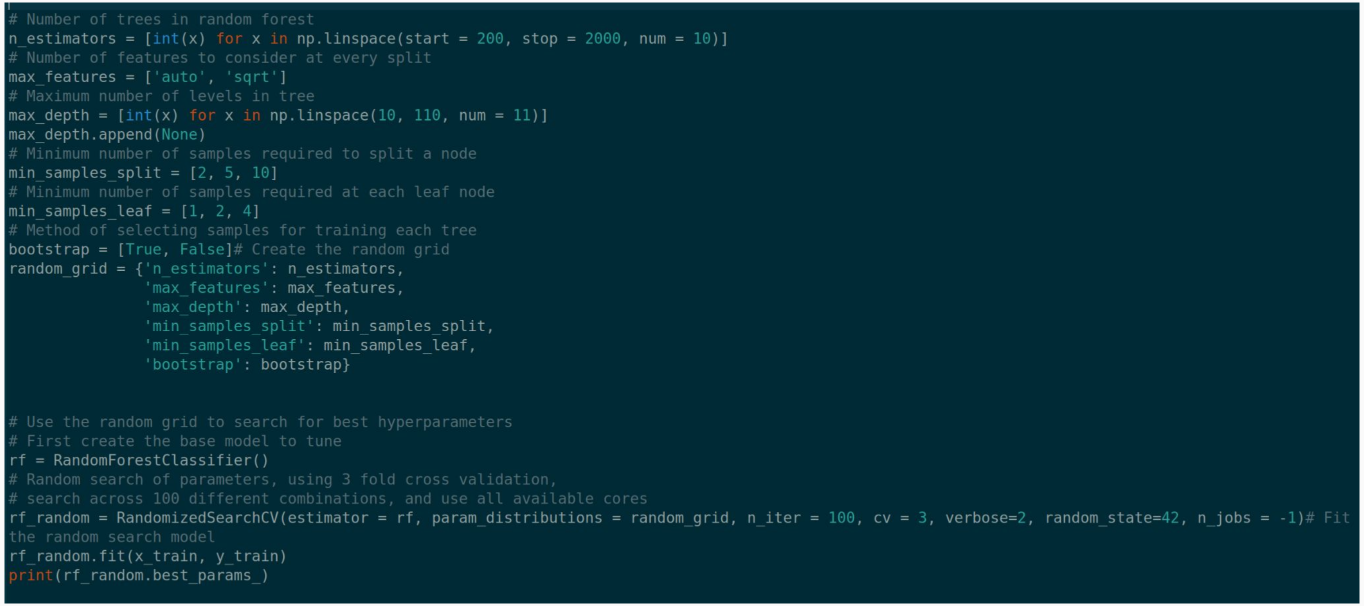
```
CatBoostClassifier:
f1_score: 0.8859725439042233
f1_score for each class: [0.90322581 0.79750779 0.918      0.96275072 0.8417787 ]
Kappa score: 0.857987730139884
Accuracy: 0.8864
Training time: 3.671531915664673

SVC:
f1_score: 0.8480497093239358
f1_score for each class: [0.84377923 0.74409044 0.8847352  0.93536122 0.8271474 ]
Kappa score: 0.8104008396391312
Accuracy: 0.8484
Training time: 31.2133150100708
```

Evaluation

	Accuracy	F1_score	Kappa_score	Training time
RandomForest	0.8828	0.8819	0.8535	8.7s
LogisticReg	0.8832	0.883	0.8540	19.5s
GradientBoosting	0.8852	0.8857	0.8565	146.5s
HistGradBoosting	0.9056	0.9052	0.8820	21.1s
LightGBM	0.908	0.9077	0.8850	1.3s
CatBoost	0.88	0.8860	0.8580	3.7s

Parameter Tuning

1. RandomForest parameters were tuned using RandomizedSearchCV



```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}


# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)# Fit
the random search model
rf_random.fit(x_train, y_train)
print(rf_random.best_params_)
```

Parameters Tuning

2. LightGBM classifier:

- Boosting algorithm was changed to “dart”
- Boosting algorithm was changed to “goss”
- Max_bins to 400 from default 255
- num_iterations increased to 200

Parameters Tuning

3. LogisticRegression

- Changed solver to liblinear(best result)
- Changed solver to other solvers(newton-cg, sag, saga)

4. Dataset preparation

- Number of samples for each group increased(500->2500)
- Number of features for each sample increased (10-1500)

Observations and improvements

1. Allergist and Dermatologist
2. Allergist and Covid
3. Y target for training from website
4. Collect better testing set for evaluating models performance