Machine Learning Nano Degree
Gender Recognition of Voice

Jiacheng Shen

May 5, 2018

Contents

1 Definition

1.1 Project Overview

Voice recognition has a long history. In 1952, Bell Lab designed a system to recongize single digit. Then, great progress was made by methods like Linear Predictive Coding, Dynamic Time Warp and Hidden Markov Model. Nowadays, Machine Learning models like RNN are applied.[18] For example, the famous Chinese company iFLYTEK can reach an accuracy of 98%.[6]

In this project, the goal is to build a model to recongize the gender of voice. The dataset can be found on Kaggle[7]. The raw wave files have been extracted features by R. There are 1584 female samples and 1584 male samples. Finally, a web service will be constructed based on the trained model as an application.

1.2 Problem Statement

This project is a **suprivised binary classification** problem. The voice can be classfied to female or male. The goal is to use the given features of voice, build a model and recongize the gender of voice.

1.3 Merrics

Since the dataset is balanced (half female and half male) and the importance of both genders are the same, so recall and precision rate mean nothing. So accuracy is chosed to evauate the model. What's more, training and predicting time is also considered because the model need to be deployed on CPU server.

1. **Accuracy**. Accuracy counts how many samples are predicted correctly.

$$\text{Accuracy} = \frac{\sum \text{Correctly predicted}}{\sum \text{All Samples}} \times 100\%$$

2. **Time**. To construct a web service on a normal CPU server, training and predicting time is also taken into account. The user cannot wait too long for the response.

2 Analysis

2.1 Data Exploration

The dataset is downloaded from Kaggle. There are 22 features and 3168 samples. It is a balanced dataset with 1584 females and 1584 males.

As shown in Table 1, there are 22 features about the voice and the last one is the label, which need to be predicted female or male.

More details about the features are shown below.[7]

1. meanfreq: mean frequency (in kHz)
2. sd: standard deviation of frequency
3. median: median frequency (in kHz)

Category	Names						
Frequency	meanfreq	sd	median	Q25	Q75	IQR	
Spectrum	skew	kurt	sp.ent	sfm	mode	centroid	peakf
Fundamental Frequency	meanfun	minfun	maxfun				
Domain Frequency	meandom	mindom	maxdom				
Range	dfrange	modindx					
output	label						

Table 1: features

4. Q25: first quantile (in kHz)
5. Q75: third quantile (in kHz)
6. IQR: interquantile range (in kHz)
7. skew: skewness (see note in specprop description)
8. kurt: kurtosis (see note in specprop description)
9. sp.ent: spectral entropy
10. sfm: spectral flatness
11. mode: mode frequency
12. centroid: frequency centroid (see specprop)
13. peakf: peak frequency (frequency with highest energy)
14. meanfun: average of fundamental frequency measured across acoustic signal
15. minfun: minimum fundamental frequency measured across acoustic signal
16. maxfun: maximum fundamental frequency measured across acoustic signal
17. meandom: average of dominant frequency measured across acoustic signal
18. mindom: minimum of dominant frequency measured across acoustic signal
19. maxdom: maximum of dominant frequency measured across acoustic signal
20. dfrange: range of dominant frequency measured across acoustic signal
21. modindx: modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
22. label: male or female

The Table 2 showes the head and the tail of the dataset. The features except label is all continuous number. The first half are all males and the second half are all females. Therefore, the dataset need to be randomly rearranged to break this distribution.

According to the Table 3, the distributions and ranges of features differ a lot from each other. Thus, normalization is needed to prevent preferrence on large features.

Features	Sample 0	Sample 1	Sample 3166	Sample 3167
meanfreq	0.059781	0.0660087	0.143659	0.165509
sd	0.0642413	0.06731	0.0906283	0.0928835
median	0.0320269	0.0402287	0.184976	0.183044
Q25	0.0150715	0.0194139	0.0435081	0.0700715
Q75	0.0901934	0.0926662	0.219943	0.250827
IQR	0.075122	0.0732523	0.176435	0.180756
skew	12.8635	22.4233	1.59106	1.70503
kurt	274.403	634.614	5.3883	5.76912
sp.ent	0.893369	0.892193	0.950436	0.938829
sfm	0.491918	0.513724	0.67547	0.601529
mode	0	0	0.212202	0.267702
centroid	0.059781	0.0660087	0.143659	0.165509
meanfun	0.0842791	0.107937	0.172375	0.185607
minfun	0.0157017	0.0158259	0.0344828	0.0622568
maxfun	0.275862	0.25	0.25	0.271186
meandom	0.0078125	0.00901442	0.79136	0.227022
mindom	0.0078125	0.0078125	0.0078125	0.0078125
maxdom	0.0078125	0.0546875	3.59375	0.554688
dfrange	0	0.046875	3.58594	0.546875
modindx	0	0.0526316	0.311002	0.35
label	male	male	female	female

Table 2: samples

	count	mean	std	min	25%	50%	75%	max
meanfreq	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
sd	3168.0	0.057126	0.016652	0.018363	0.041954	0.059155	0.067020	0.115273
median	3168.0	0.185621	0.036360	0.010975	0.169593	0.190032	0.210618	0.261224
Q25	3168.0	0.140456	0.048680	0.000229	0.111087	0.140286	0.175939	0.247347
Q75	3168.0	0.224765	0.023639	0.042946	0.208747	0.225684	0.243660	0.273469
IQR	3168.0	0.084309	0.042783	0.014558	0.042560	0.094280	0.114175	0.252225
skew	3168.0	3.140168	4.240529	0.141735	1.649569	2.197101	2.931694	34.725453
kurt	3168.0	36.568461	134.928661	2.068455	5.669547	8.318463	13.648905	1309.612887
sp.ent	3168.0	0.895127	0.044980	0.738651	0.861811	0.901767	0.928713	0.981997
sfm	3168.0	0.408216	0.177521	0.036876	0.258041	0.396335	0.533676	0.842936
mode	3168.0	0.165282	0.077203	0.000000	0.118016	0.186599	0.221104	0.280000
centroid	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
meanfun	3168.0	0.142807	0.032304	0.055565	0.116998	0.140519	0.169581	0.237636
minfun	3168.0	0.036802	0.019220	0.009775	0.018223	0.046110	0.047904	0.204082
maxfun	3168.0	0.258842	0.030077	0.103093	0.253968	0.271186	0.277457	0.279114
meandom	3168.0	0.829211	0.525205	0.007812	0.419828	0.765795	1.177166	2.957682
mindom	3168.0	0.052647	0.063299	0.004883	0.007812	0.023438	0.070312	0.458984
maxdom	3168.0	5.047277	3.521157	0.007812	2.070312	4.992188	7.007812	21.867188
dfrange	3168.0	4.994630	3.520039	0.000000	2.044922	4.945312	6.992188	21.843750
modindx	3168.0	0.173752	0.119454	0.000000	0.099766	0.139357	0.209183	0.932374

Table 3: describe

2.2 Exploratory Visualization

The figure 1 shows the histogram of each feature. The features **IQR**, **meanfun** and **sd** have two peak, which may mean the difference between the voice of female and male.

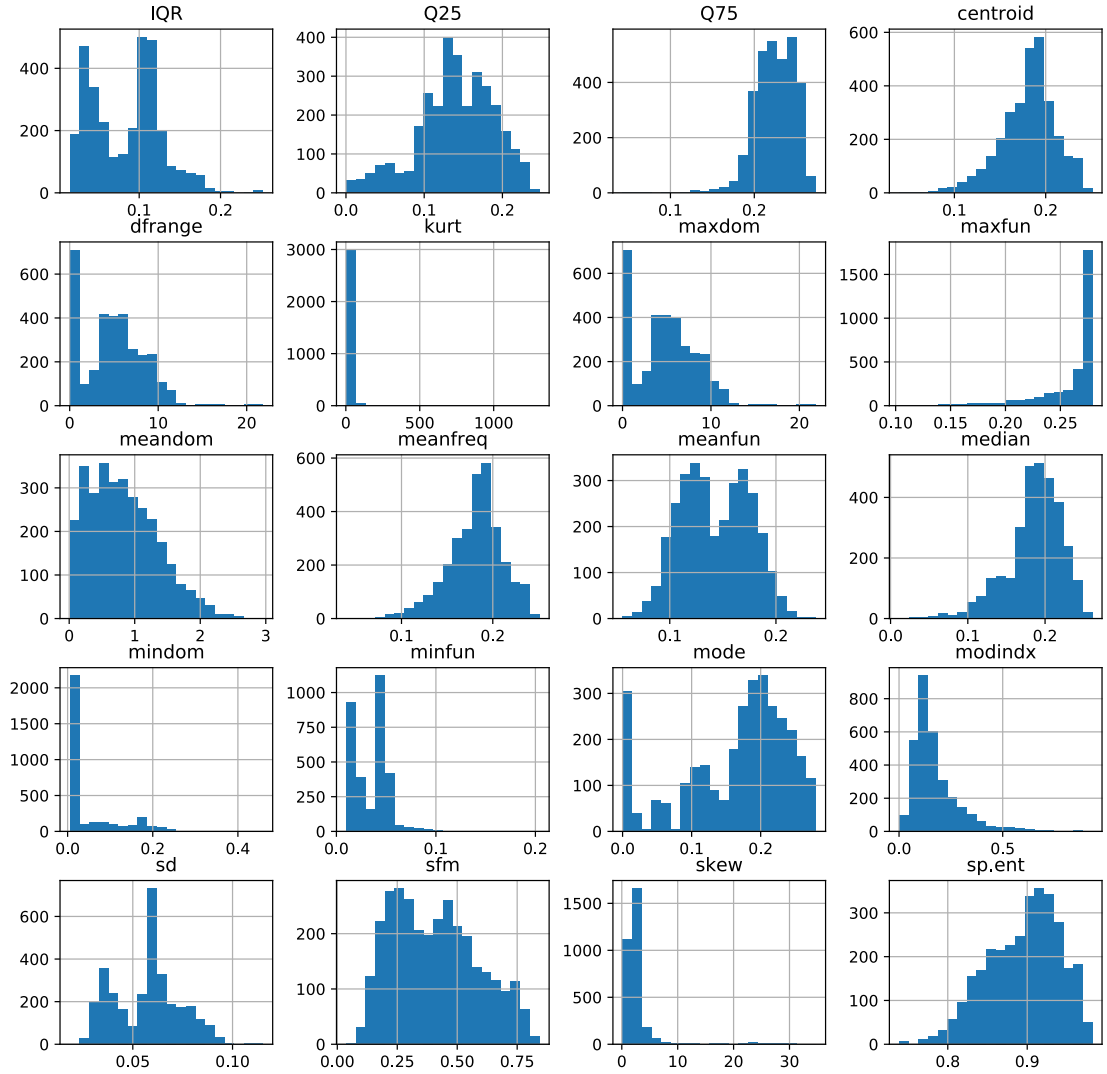

Figure 1: Distribution of Features

2.3 Algorithms and Techniques

I use sklearn package from python to do this project. The features of dataset are all numbers, so no encoding is needed. The label has only two cases, so converting it into a binary number is enought. Some features have some skewness, so the log transformation need to be applied. The dataset is balanced, but the first half are all males, so I will use train_test_split with random state to split Training sets, Validation sets and Testing sets. Since it is a binary classification problem, I choose

LogisticRegression, DecisionTree, RandomForest and SVM as candidates. After choosing the model with the best performance, I will use GridSearch to optimize this model.

1. LogisticRegression:

    - Pros: LogisticRegression trys to find a line fit the data best with least error. The model can be hardly over-fitting. Thanks to Calculus, the parameters can be calculated fast. So this model trains and predicts fast. What's more, it can give us the correlation between features and label in the form of weights.
    - Cons: It can only express linear correlation, so if the data has nolinear relations, it doesn't work well. Even if it is possible to convert the complex relations to some kind of linear correlation, it is hard to choose the transforming function. It is also easy to be under-fitted. If the dataset has large feature spaces, it is hard for LogisticRegression to handle so many features and it works poor, too.

2. DecisionTree

    - Pros: DecisionTree trys to divide situations by different range of features. It can express complex relations, more than linear relations. It generates a tree structure, so it trains and predicts fast. The trained model is easy to explain from the standards to divide. Even if some features are lost, DecisionTree can ignore it and train the model.
    - Cons: It is easy to be over-fitting if the tree trys to fit every train sample.

3. RandomForest

    - Pros: RandomForest combines lots of tree trained on part of training set. Even if the feature space is huge, it can split them into smaller one and train on a tree, so it can handle huge feature space. The trained forest is easy to explain from the trees and standards of each tree. If some features are lost, it can also work like DecisionTree.
    - Cons: If the dataset has too much noise, RandomForest is also easy to be over-fitting. It costs much time if there are many estimators.

4. SVM

    - Pros: SVM trys to find a line spliting the samples and maximuming the distances. So it has good generalization performance. Using kernel functions, it can also work on nolinear correlation.
    - Cons: It is sensible to the kernel function. An unproper kernel function leads to poor performance. It trains slow because it needs do a lot of calculation to maximum the distance. The trained model is hard to explain since its formulas have few direct meanings.

2.4 Benchmark

Using a simple model like LogisticRegression(shown in the jupyter notebook of the repo), I can achieve accuarcy of 90.96% on the training dataset and 90.38% on the testing dataset. In the webpage of Kaggle[7], they can achieve 97% / 98% accuracy by Logistic Regression, 100% / 98% by Random Forest and 100% / 99% by SVM(Sorry there is no clear explanation of what "xx%/xx%" means. I suppose it is the accuracy of female and male on the whole dataset). I don't have powerful computer to try all the parameters in the GridSearch, so I should try to perform better than the simple model and be close to the accuarcy on the webpage. I want to achieve bias and variance as followings:

1. Bias: The ideal model should predict each voice to the right gender. I set the accuarcy threshold as 97.5% on the **Testing** dataset.

2. Variances: The good model should perform similar on different dataset to avoid over-fitting. I limit the difference between Training and Testing dataset to less than 1%.

3 Methodology

3.1 Data Preprocessing

The features are numbers so no one-hot encoding is needed. The label has only two values, 'female' and 'male', so encoding it into binary is enough. Some features like 'kurt', 'drange' and 'skew' have kind of skewness, so implementing log transformation on them ,shown on 2.

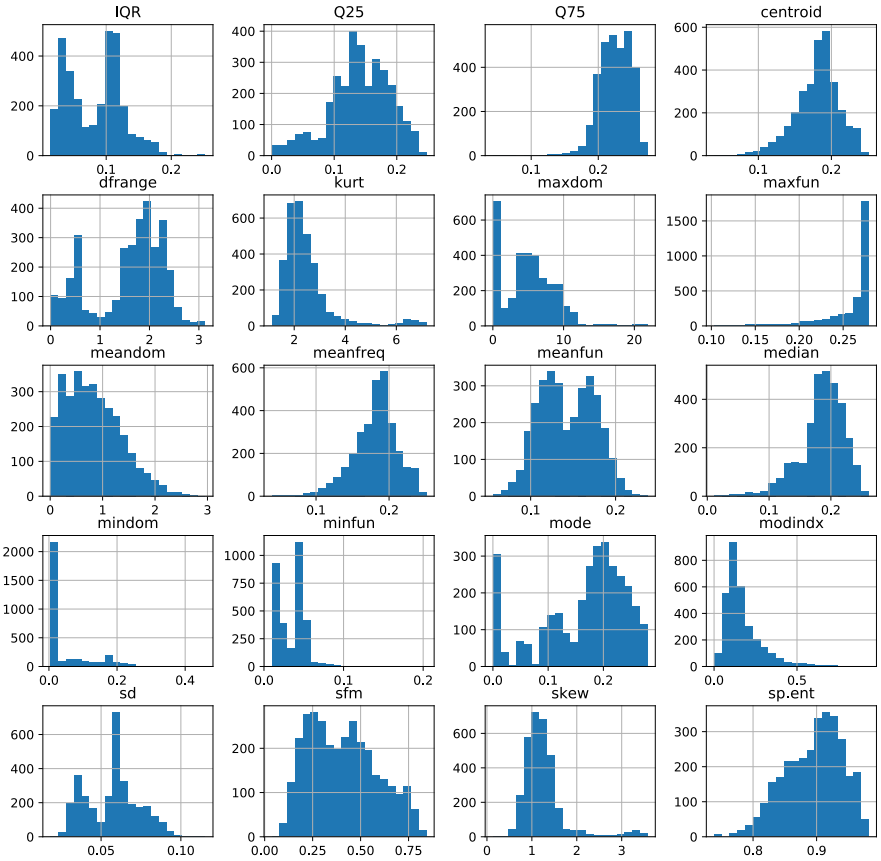

Figure 2: Distribution of log Features

The ranges of features differs from each other, so normalization is implemented for further model training. I use MinMaxScaler from sklearn.preprocessing to scale the features to [0,1]. Finally the dataset is splited into testing set, training set and validation set with shuffling. I use train_test_split from sklearn.model_selection to split the dataset. The spilted dataset is shown in table 4.

Dataset	Size
Train	2027
Validation	507
Test	634

Table 4: splited dataset

3.2 Implementation

I tried LogisticRegression, DecisionTree, RandomForest and SVM:

1. For LogisticRegression, it is imported from sklearn.linear_model. Random_state is set for reappearing. All other parameters are default: penalty='l2', dual = False, tol=1e-4, C=1.0, class_weight=None, solver='liblinear'.

2. For DecisionTree, it is imported from sklearn.tree. Random_state is set for reappearing. All other parameters are default: criterion='gini', splitter='best', max_depth=None, min_samples_split= min_samples_leaf=1, max_features=None, class_weight=None.

3. For RandomForest, it is imported from sklearn.ensemble. Random_state is set for reappearing. All other parameters are default: criterion='gini', max_features="auto", max_depth=None, min_samples_split=2, min_samples_leaf=1, bootstrap=True, class_weight=None.

4. For SVM, it is imported from sklearn.svm. Random_state is set for reappearing. All other parameters are default: C=1.0, kernel='rbf', gamma='auto', shrinking=True, tol=1e-3, decision_function_shape='ovr'.

The comparsion of four models on aspects of **Training Time**, **Predicting Time**, **Training Accuracy** and **Validation Accuracy** is shown in figure 3.

1. SVM has small gap between train accuarcy and test accuracy, which means it may have good generalization performance. However, it costs too much time to train and predict, and it doesn't achieve the highest Validation Accuarcy.

2. LogisticRegression and DecisionTree cost little time, but they don't achieve highest accuracy on Validation set.

3. RandomForest achieve highest Validation Accuracy and costs acceptable training and predicting time.

RandomForest performs well with default parameters. However, no model achieve $> 97.5\%$ validation accuarcy. Thus, model refinement is needed.

3.3 Refinement

I use GridSearch from sklearn.model_selection with 4-Fold to optimize the model with following parameter combinations, shown in table 5:

According to document from sklearn [13][16][12][15], these parameters have such meanings:

1. C: For LogisticRegression, it means the inverse of regularization strength; for SVC, it means the penalty of the error term. In general, it controls how good should the model fit the training dataset, default 1.0. If it fit the training dataset too well, it may lead to over-fitting. So I try 0.5, 1.0 and 2.0 to choose the best one.

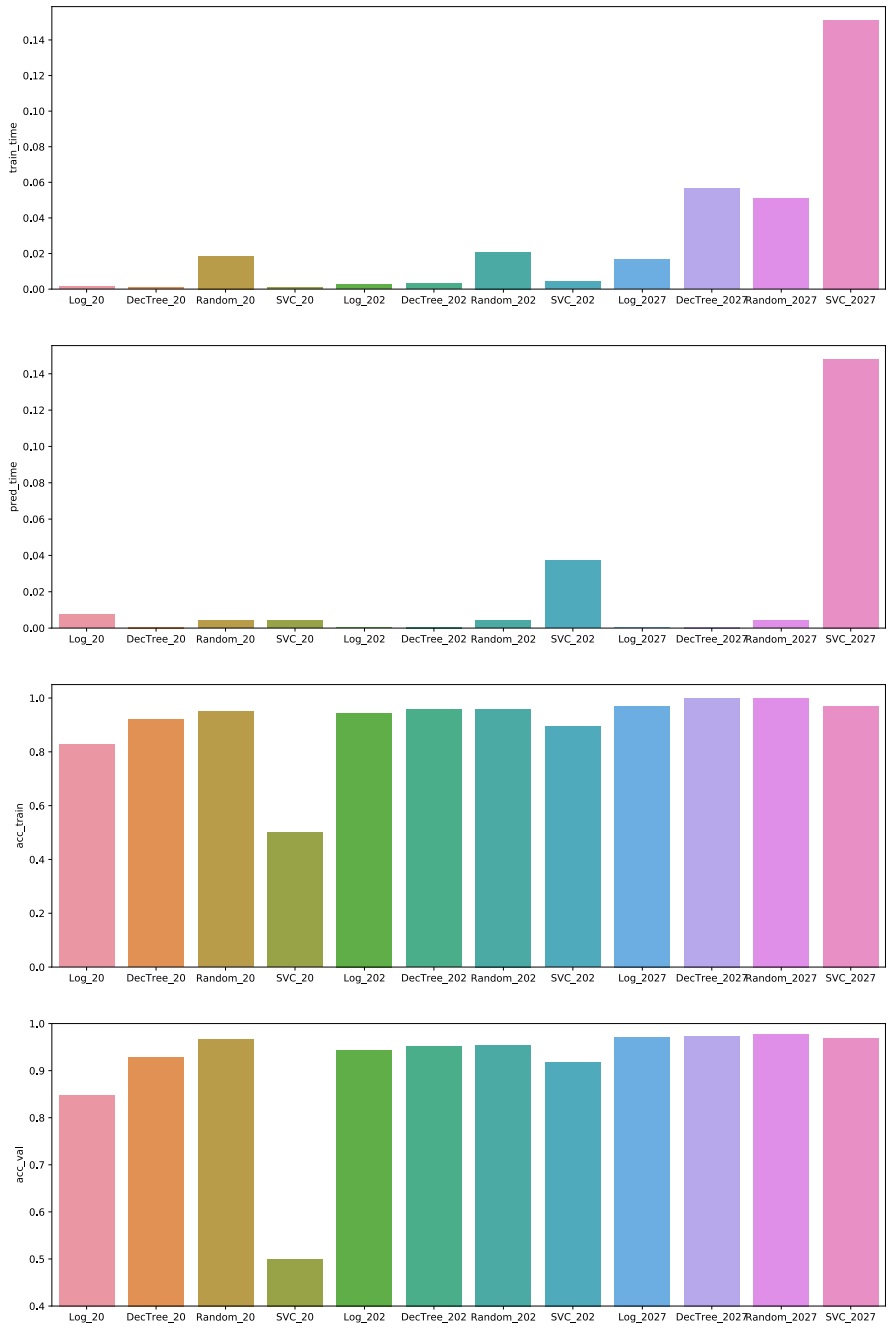

Figure 3: Comparsion of four models

Model	Parameter	Value			
LogisticRegression	C	0.5	1.0	2.0	
	solver	"liblinear"	"sag"	"saga"	
DecisionTree	criterion	"gini"	"entropy"		
	max_depth	8	10	12	
RandomForest	n_estimators	50	100	200	400
	criterion	"gini"	"entropy"		
	max_depth	8	10	12	
SVC	C	0.1	0.5	1.0	
	gamma	1	3	5	7

Table 5: features

2. solver: For LogisticRegression, it is the Algorithm to use in the optimization problem. I try liblinear, sag and saga.

3. criterion: It is a string, default "gini". For DecisionTree and RandomForest, it means the function to measure the quality of a split. "gini" is the Gini impurity and "entropy" is the information gain. There are total two candidates so it is fine to try both.

4. max_depth: For DecisionTree and RandomForest, it means the max depth of the tree. It prevent the tree growing too deeply to avoid over-fitting. I try 5, 8, 10, 12.

5. n_estimators: It is a integer, default 10. It means the number of trees in the forest. Since RandomForest is an ensemble model, the more trees in the forest, the more accuarcy it may achieve and the less possibility that it may be over-fitting. However, too many trees will also cause computing performance problems. Thus, I set upbound to 400 for the balance and try 50, 100, 200, 400.

6. gamma: For SVC, it means kernel coefficient. It works for kernel 'rbf'(default kernel), 'poly' and 'sigmoid'. By default it is 1/n_features and it doesn't work well in figure 3. I try 1, 3, 5, 7.

After running GridSearch for about ten minutes on my laptop, the tuned parameters are shown in table 6.

Model	Parameter	Tuned_value
LogisticRegression	C	2.0
	solver	"liblinear"
DecisionTree	criterion	"entropy"
	max_depth	8
RandomForest	n_estimators	100
	criterion	"entropy"
	max_depth	12
SVC	C	1.0
	gamma	5

Table 6: tuned parameters

The comparsion of four optimized models on aspects of **Training Time**, **Predicting Time**, **Training Accuracy** and **Validation Accuracy** is shown in figure 4.

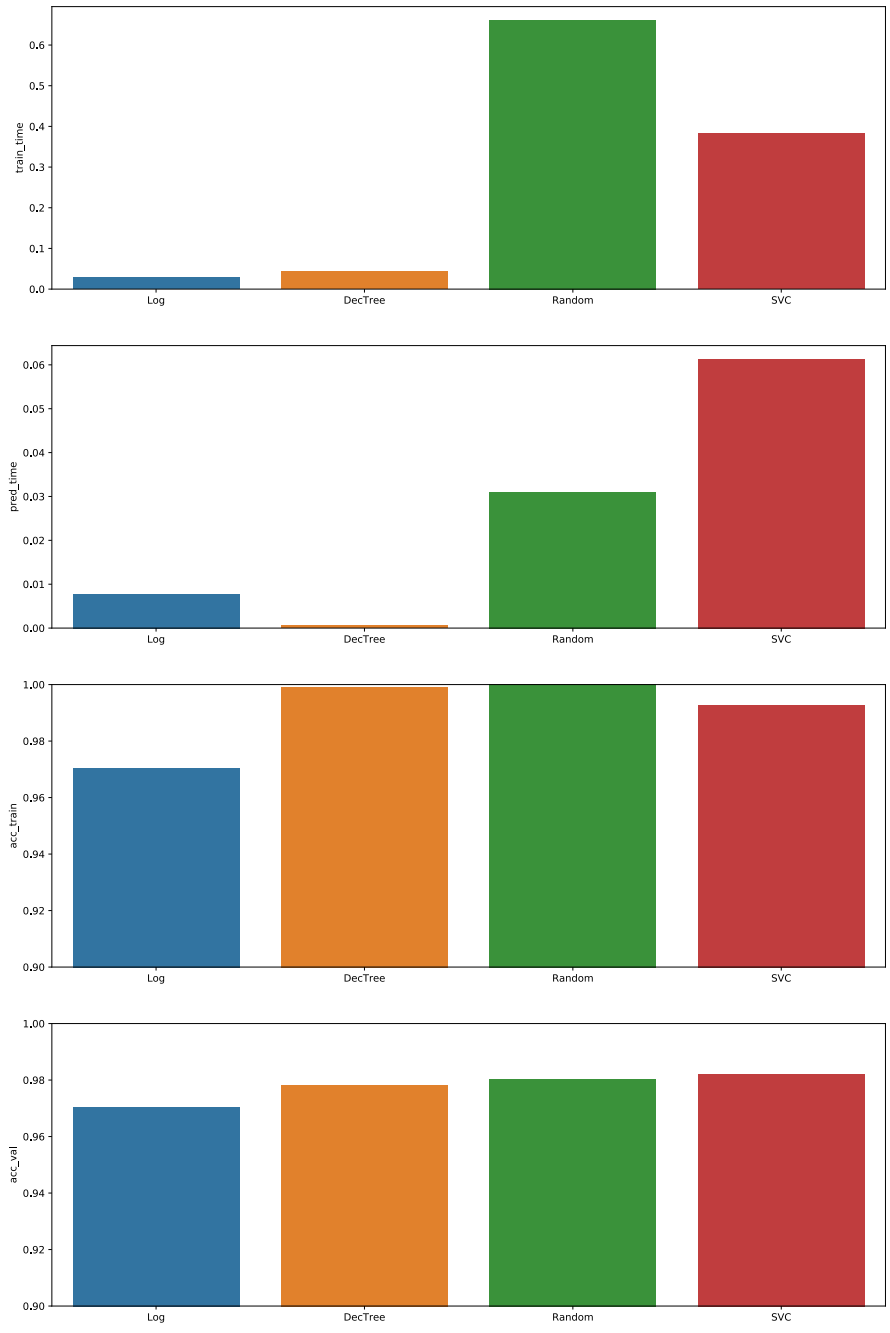

Figure 4: Comparsion of four optimized models

1. LogisticRegression costs least time and have similar accuracy on training and validation dataset. However, it can only achieve accuarcy of about 97.0%. This dataset may be beyond the relations it can handle.

2. DecisionTree costs little time. It performs well on the training dataset, almost 100%. But it can only achieve about 98% on the validation dataset. The gap of 2% means over-fitting happens even if the max_depth is restricted. Its performance on testing data is doubtful.

3. Similiar to DecisionTree, it achieves exactly 100% accuarcy on training dataset but 98% on validation dataset. Over-fitting happens, too. What's more, it costs most time in training.

4. SVC achieves highest accuarcy on validation dataset. And the gap between training and validation dataset is smaller than DecisionTree and RandomForest. Even if it costs most time to predict, the predicting time is still acceptable, 0.05 second.

3.4 Improvement

Although SVC performs well both on training and validation dataset, there is still a gap larger than 1%. So I will try to select several most important features to improve the generalization performance.

I use SVC with linear kernel to calculate the feature importance [1]. Then, try to train the optimized model on different numbers of important features. The result is shown in figure 5.

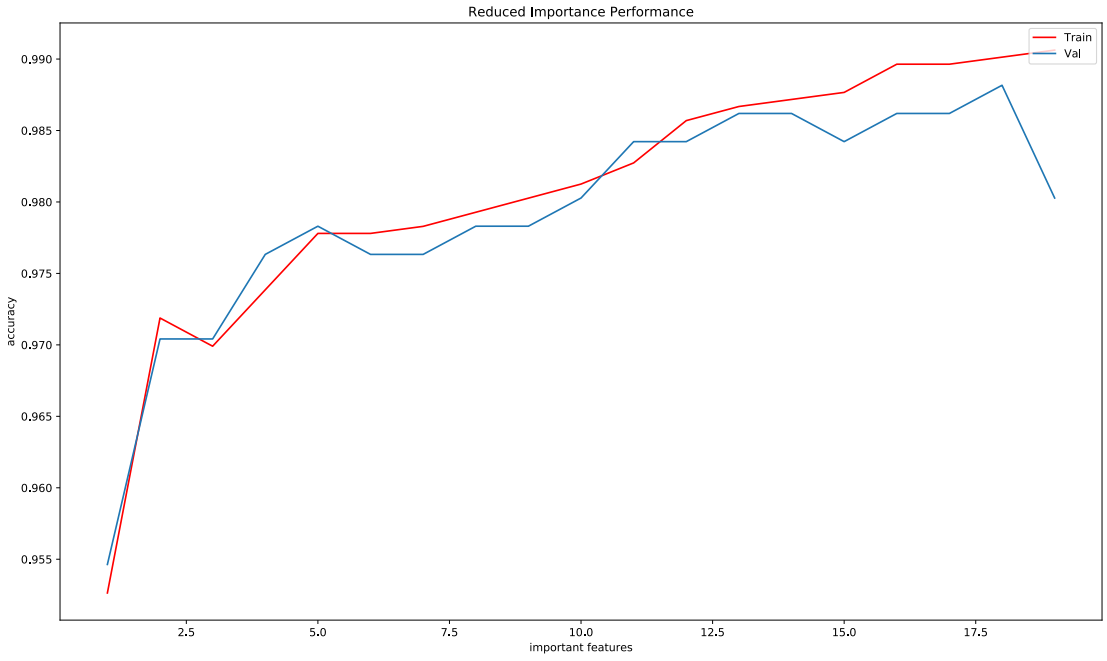

Figure 5: Accuracy on part important features

Among the points above 98% accuracy, the optimized model trained on 13 most important features has the smallest gap(0.05%) between training(98.67%) and validation accuracy(98.62%). Thus, we can easily infer that this model has large possibility to perform well on testing dataset, too.

4 Results

4.1 Model Evaluation and Validation

Now, let's compare each model's performance on training, validation and testing dataset in table 7.

Model	Train	Validation	Test	Train Time	Predict Time
LogisticRegression	97.04%	97.04%	96.37%	0.0289	0.0078
DecisionTreeClassifier	99.90%	97.83%	96.53%	0.0431	0.0006
RandomForestClassifier	100.00%	98.03%	97.95%	0.6614	0.0310
SVC	99.26 %	98.22%	98.11%	0.3816	0.0613
reduced SVC	98.67%	98.62%	97.95%	0.2682	0.0074

Table 7: Final Model Compare

4.2 Justification

RandomForestClassifier, SVC and reduced SVC have testing accuracy larger than 97.5%. However, RandomForestClassifier and SVC have gap of more than 1% between training and testing accuracy while reduced SVC has gap of 0.72%.

What's more, the reduced SVC costs much less training and predicting time than RandomForestClassifier and SVC.

Therefore, the reduced SVC is the final model. It can achieve the high accuracy more than 97.5% on testing dataset and has a small gap less than 1% between training and testing accuarcy.

5 Conclusion

5.1 Free-Form Visualization

Let's check which features are most important in figure 6:

The top-13 important features are

1. meanfun
2. IQR
3. Q75
4. minfun
5. sfm
6. sp.ent
7. Q25
8. modindx
9. skew
10. median
11. sd

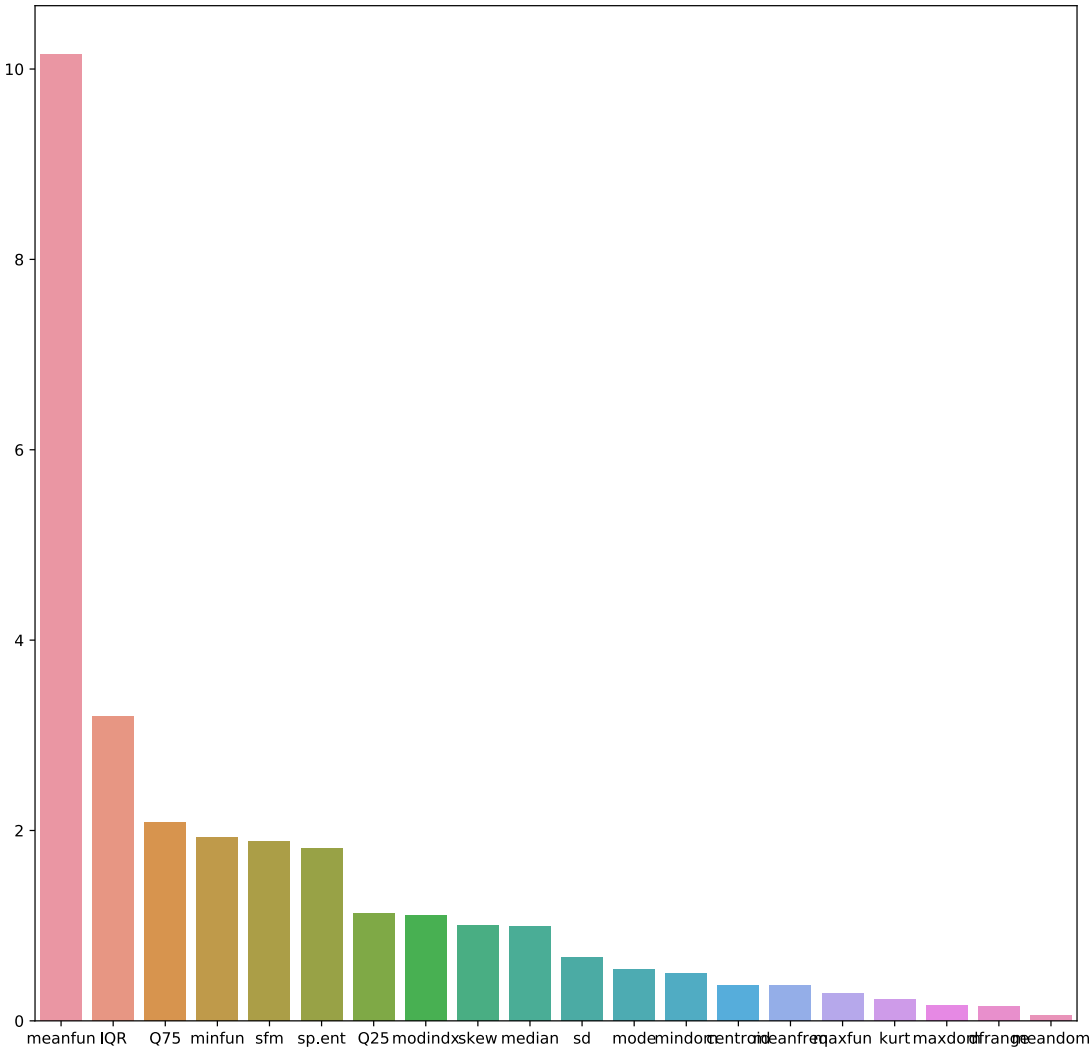

Figure 6: Feature Importance

12. mode
13. mindom

And 3 of them, **IQR**, **meanfun**, **sd** matches observation before in **Exploratory Visualization**.

5.2 Reflection

RandomForest achieves 100% accuracy on the training dataset, but it cannot perform well on the validation and testing dataset, too. Obviously, over-fitting happens. The SVM trained on full training data has the highest accuarcy on testing dataset. However, there is still a gap larger than 1% between training accuarcy and testing accuracy. Although the reduced SVM has the same testing accuarcy as RandomForest, it performs similar on every dataset, so we can infer that it **does** find the common rules of this dataset.

6 Application

I deploy a web service to recongize the uploaded voice. I choose Python as the main language to implement it. After searching, I use Flask[4] as the framework, use rpy2[11] to extract features from raw voice by R packages.

The source code is in folder web. To deploy it, I search and solve the following problems.

1. How to upload file using flask [5]
2. The returned values of fund [10]
3. How to scale on reduced feature [14]
4. How to convert voice in to 'wav' type using ffmpeg [8]
5. The beautiful html template [2]
6. How to change css in flask [3]
7. How to customize upload button [17]
8. How to deploy flask with gunicorn and nginx [9]

The web service is deployed on http://qasd.tk:4040

7 Reference

References

[1] Aneesha Bakharia. Visualising top features in linear svm with scikit learn and matplotlib. https://medium.com/@aneesha/visualising-top-features-in-linear-svm-with-scikit-learn-and-matplotlib-3454ab18a14d.

[2] CodyHouse. Animated intro section. https://codyhouse.co/gem/animated-intro-section/.

[3] feesland. Flask . http://www.cnblogs.com/feeland/p/4640695.html.

[4] flask. Flask (a python microframework). http://flask.pocoo.org/.

[5] Flask. Uploading files. http://flask.pocoo.org/docs/0.12/patterns/fileuploads.

[6] iFLYTEK. iflytek open platform. http://www.xfyun.cn/.

[7] Kaggle. Gender recognition of voice. https://www.kaggle.com/primaryobjects/voicegender.

[8] martineau. Convert mp3 to wav using ffmpeg for vbr. https://superuser.com/questions/675342/convert-mp3-to-wav-using-ffmpeg-for-vbr.

[9] Melwood. Flask + gunicorn + nginx . https://jiayi.space/post/flask-gunicorn-nginx-bu-shu.

[10] R. fund function. https://www.rdocumentation.org/packages/seewave/versions/2.0.5/topics/fund.

[11] rpy2. R in python. https://rpy2.bitbucket.io/.

[12] sklearn. sklearn.ensemble.randomforestclassifier. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[13] sklearn. sklearn.linear_model.logisticregression. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[14] sklearn. sklearn.preprocessing.minmaxscalar. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[15] sklearn. sklearn.svm.svc. http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[16] sklearn. sklearn.tree.decisiontreeclassifier. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[17] Osvaldas Valutis. Styling & customizing file inputs the smart way. https://tympanus.net/codrops/2015/09/15/styling-customizing-file-inputs-smart-way.

[18] Wikipedia. Speech recognition. https://en.wikipedia.org/wiki/Speech_recognition#History.