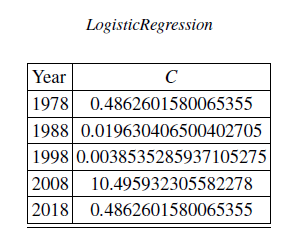
**Appendix**

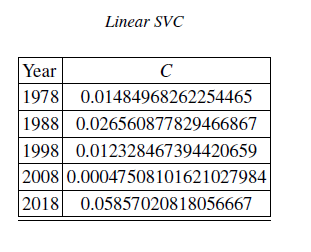
1. **Hyperparameters**

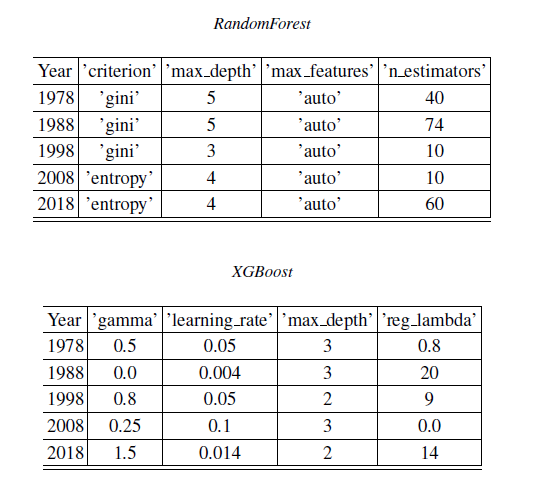
We fine-tuned the models by using *GridSearchCV* to find best fitting parameters. Taking the multiclass logistic regression model as an example:

First, we searched for the hyperparameter *C* from the list numpy.logspace(-4, 4, 100) and found the best one is 0.5689866029018293. We then explored the parameter space around value 0.5689866029018293: np.logspace(-1, 0, 100). We found the best one is 0.4862601580065355. As further tuning turned out not affecting the f1 score more than 1%, we terminated the process.

The final hyperparameters used for all saturated models are listed below.





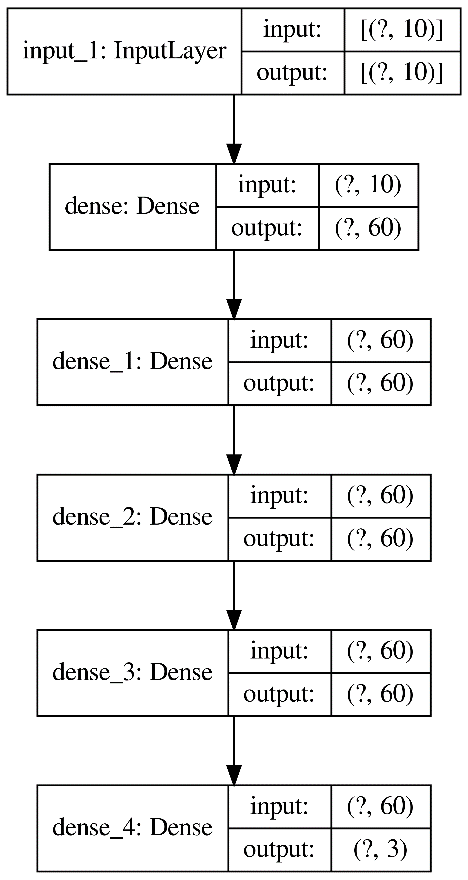


For the DNN models, we constructed the dynamic DNN structures using Subclassing API from Keras. Since our goal was not to find the best performing model, we only studied the simple Sequential structures. Since Keras does not provide weighted f1 score as its stand loss function or standard metric, we must implement them as customized functions. No that the metric will compute the f1 score, but the loss function shall return 1-f1\_score.

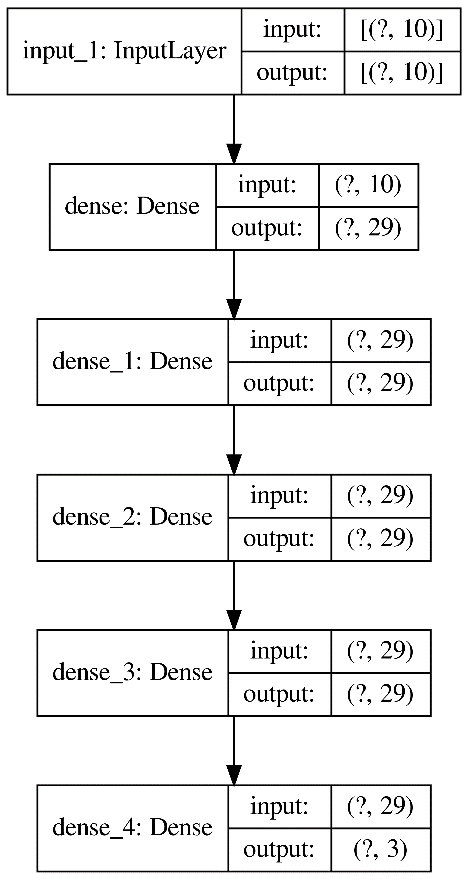
We used 80% of the training instances for training and 20% as validation set for early stop, where we set the patience to 10.

The DNN structures are displayed below.

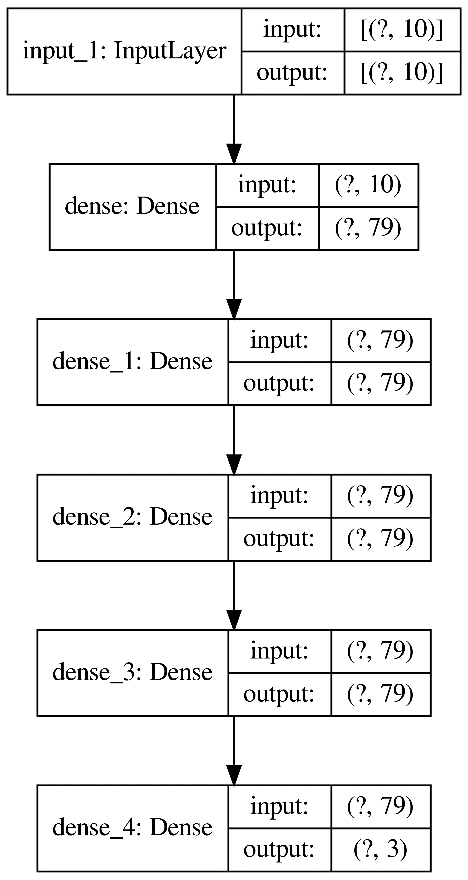
1. **1978**



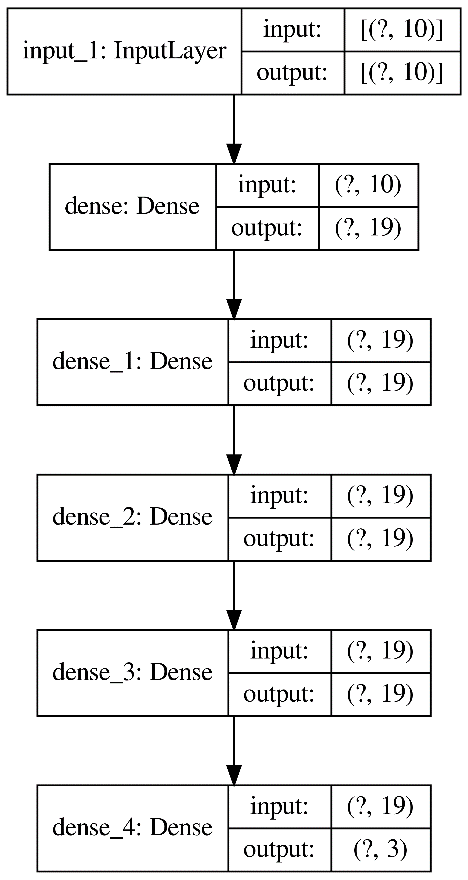
1. **1988**



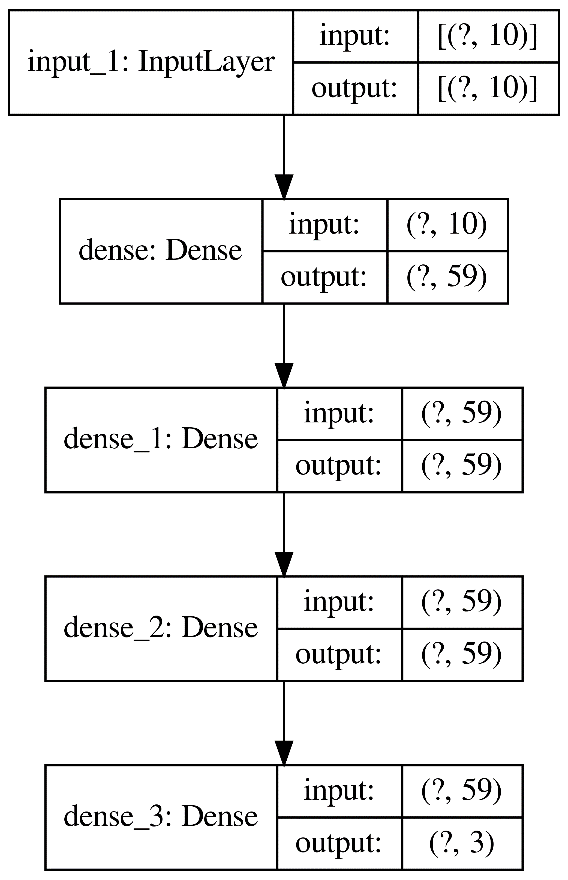
1. **1998**



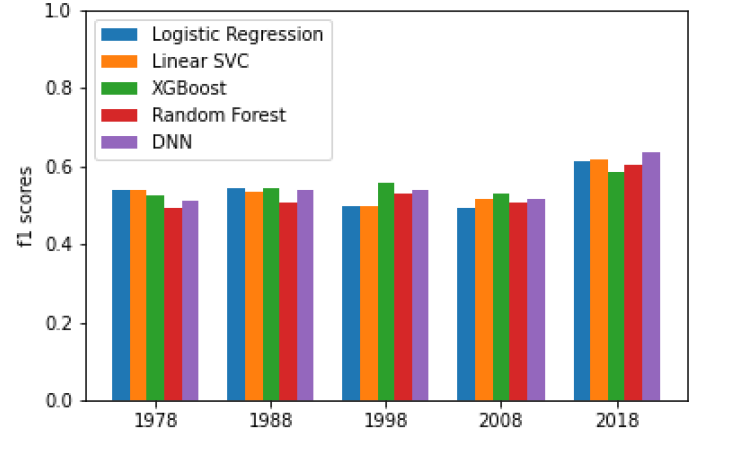
1. **2008**



1. **2018**



1. **F1 scores**



1. **Shapley values from other models**

We reported the Shapley values generated from other models in the following figures.

1. **Linear SVC**

Chart, bar chart

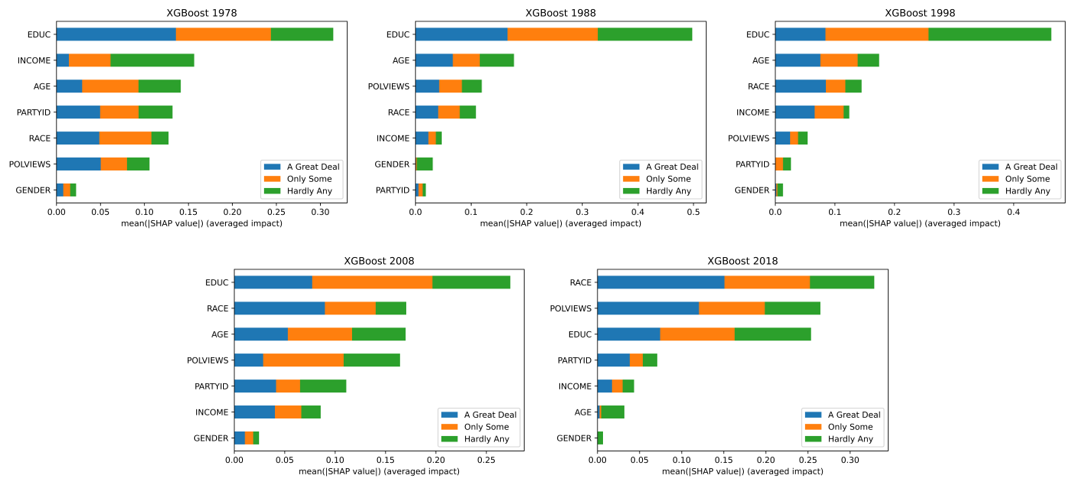
Description automatically generated

1. **Random Forest**

Chart, bar chart

Description automatically generated

1. **XGBoost**



1. **DNN**

Chart, bar chart

Description automatically generated