```
import pandas as pd
train=pd.read_csv('Train_data.csv')
train.head()
```

	a1	a2	a3	a4	a5	a6	b1	b2	b3	b4	...	f4	f5	f6	g1	g2	g3	g4	g5	g6	class
0	b	b	b	b	b	b	o	o	x	b	...	b	b	b	b	b	b	b	b	b	loss
1	b	b	b	b	b	b	o	b	b	b	...	b	b	b	b	b	b	b	b	b	loss
2	x	x	o	b	b	b	o	b	b	b	...	b	b	b	b	b	b	b	b	b	win
3	x	x	b	b	b	b	o	b	b	b	...	b	b	b	o	x	o	x	b	b	win
4	x	x	o	o	b	b	o	b	b	b	...	b	b	b	b	b	b	b	b	b	win

5 rows × 43 columns

```
test=pd.read_csv('Test_data.csv.csv')
test.head()
```

	a1	a2	a3	a4	a5	a6	b1	b2	b3	b4	...	f3	f4	f5	f6	g1	g2	g3	g4	g5	g6
0	b	b	b	b	b	b	b	b	b	b	...	b	b	b	b	x	x	o	b	b	b
1	b	b	b	b	b	b	x	o	o	o	...	b	b	b	b	x	b	b	b	b	b
2	x	x	o	b	b	b	x	o	b	b	...	b	b	b	b	o	b	b	b	b	b
3	b	b	b	b	b	b	x	o	o	b	...	b	b	b	b	o	x	x	b	b	b
4	x	o	b	b	b	b	b	b	b	b	...	b	b	b	b	x	x	o	b	b	b

5 rows × 42 columns

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50667 entries, 0 to 50666
Data columns (total 43 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   a1      50667 non-null  object
 1   a2      50667 non-null  object
 2   a3      50667 non-null  object
 3   a4      50667 non-null  object
 4   a5      50667 non-null  object
 5   a6      50667 non-null  object
 6   b1      50667 non-null  object
 7   b2      50667 non-null  object
 8   b3      50667 non-null  object
 9   b4      50667 non-null  object
 10  b5      50667 non-null  object
 11  b6      50667 non-null  object
 12  c1      50667 non-null  object
 13  c2      50667 non-null  object
 14  c3      50667 non-null  object
 15  c4      50667 non-null  object
 16  c5      50667 non-null  object
 17  c6      50667 non-null  object
 18  d1      50667 non-null  object
 19  d2      50667 non-null  object
 20  d3      50667 non-null  object
 21  d4      50667 non-null  object
 22  d5      50667 non-null  object
 23  d6      50667 non-null  object
 24  e1      50667 non-null  object
 25  e2      50667 non-null  object
 26  e3      50667 non-null  object
 27  e4      50667 non-null  object
 28  e5      50667 non-null  object
 29  e6      50667 non-null  object
 30  f1      50667 non-null  object
 31  f2      50667 non-null  object
 32  f3      50667 non-null  object
 33  f4      50667 non-null  object
 34  f5      50667 non-null  object
 35  f6      50667 non-null  object
 36  g1      50667 non-null  object
 37  g2      50667 non-null  object
 38  g3      50667 non-null  object
```

```
 39  g4      50667 non-null  object
 40  g5      50667 non-null  object
 41  g6      50667 non-null  object
 42  class   50667 non-null  object
dtypes: object(43)
memory usage: 16.6+ MB
```

```
train.nunique()
```

```
a1       3
a2       3
a3       3
a4       3
a5       3
a6       3
b1       3
b2       3
b3       3
b4       3
b5       3
b6       3
c1       3
c2       3
c3       3
c4       3
c5       3
c6       3
d1       3
d2       3
d3       3
d4       3
d5       3
d6       3
e1       3
e2       3
e3       3
e4       3
e5       3
e6       3
f1       3
f2       3
f3       3
f4       3
f5       3
f6       3
g1       3
g2       3
g3       3
g4       3
g5       3
g6       3
class    3
dtype: int64
```

```
round(train['class'].value_counts()*100/len(train),2)
```

```
win     65.83
loss    24.62
draw     9.55
Name: class, dtype: float64
```

```
!pip install flaml
from sklearn.metrics import mean_absolute_percentage_error
```

```
Collecting flaml
  Using cached FLAML-2.1.0-py3-none-any.whl (295 kB)
Requirement already satisfied: NumPy>=1.17.0rc1 in /usr/local/lib/python3.10/dist-packages (from flaml) (1.23.5)
Installing collected packages: flaml
Successfully installed flaml-2.1.0
```

```
from flaml import AutoML
automl = AutoML()
```

```
y = train.pop('class')
X = train
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=42,shuffle=True, stratify=y)
```

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
```

```
X_train.shape, y_train.shape
```

```
((40533, 42), (40533,))
```

```
X_train=pd.get_dummies(X_train,prefix_sep='__')
X_train.head()
```

	a1__b	a1__o	a1__x	a2__b	a2__o	a2__x	a3__b	a3__o	a3__x	a4__b	...	g3__x	g4__b	g4__o	g4__x	g5__b	g5__o	g
45705	0	0	1	1	0	0	1	0	0	1	...	0	1	0	0	1	0	
5889	1	0	0	1	0	0	1	0	0	1	...	0	1	0	0	1	0	
26479	1	0	0	1	0	0	1	0	0	1	...	0	1	0	0	1	0	
26889	0	0	1	1	0	0	1	0	0	1	...	0	1	0	0	1	0	
31487	0	1	0	1	0	0	1	0	0	1	...	0	1	0	0	1	0	

5 rows × 126 columns

```
X_train, y_train = sm.fit_resample(X_train, y_train)
X_train.shape, y_train.shape
```

```
((80049, 126), (80049,))
```

```
def undummify(df, prefix_sep="__"):
    cols2collapse = {
        item.split(prefix_sep)[0]: (prefix_sep in item) for item in df.columns
    }
    series_list = []
    for col, needs_to_collapse in cols2collapse.items():
        if needs_to_collapse:
            undummified = (
                df.filter(like=col)
                .idxmax(axis=1)
                .apply(lambda x: x.split(prefix_sep, maxsplit=1)[1])
                .rename(col)
            )
            series_list.append(undummified)
        else:
            series_list.append(df[col])
    undummified_df = pd.concat(series_list, axis=1)
    return undummified_df
```

```
X_train=undummify(X_train)
X_train.head()
```

	a1	a2	a3	a4	a5	a6	b1	b2	b3	b4	...	f3	f4	f5	f6	g1	g2	g3	g4	g5	g6
0	x	b	b	b	b	b	o	o	b	b	...	b	b	b	b	b	b	b	b	b	b
1	b	b	b	b	b	b	b	b	b	b	...	b	b	b	b	o	x	b	b	b	b
2	b	b	b	b	b	b	x	o	b	b	...	b	b	b	b	x	b	b	b	b	b
3	x	b	b	b	b	b	b	b	b	b	...	b	b	b	b	b	b	b	b	b	b
4	o	b	b	b	b	b	o	b	b	b	...	b	b	b	b	b	b	b	b	b	b

5 rows × 42 columns

```
automl.fit(X_train, y_train, task="classification",metric='log_loss',time_budget=900)
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/sklearn.py:1395: UserWarning: `use_label_encoder` is deprecated in 1.7.0.
  warnings.warn("`use_label_encoder` is deprecated in 1.7.0.")
[flaml.automl.logger: 09-23 16:40:34] {2391} INFO -  at 340.7s, estimator xgb_limitdepth's best error=0.5576,   best estimator lgbm'
[flaml.automl.logger: 09-23 16:40:34] {2218} INFO - iteration 111, current learner rf
[flaml.automl.logger: 09-23 16:40:35] {2391} INFO -  at 341.7s, estimator rf's best error=0.6371,       best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:40:35] {2218} INFO - iteration 112, current learner lgbm
[flaml.automl.logger: 09-23 16:41:17] {2391} INFO -  at 383.5s, estimator lgbm's best error=0.0899,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:41:17] {2218} INFO - iteration 113, current learner extra_tree
[flaml.automl.logger: 09-23 16:41:17] {2391} INFO -  at 383.9s, estimator extra_tree's best error=0.7052,       best estimator lgbm'
[flaml.automl.logger: 09-23 16:41:17] {2218} INFO - iteration 114, current learner rf
[flaml.automl.logger: 09-23 16:41:19] {2391} INFO -  at 385.3s, estimator rf's best error=0.6371,       best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:41:19] {2218} INFO - iteration 115, current learner lrl1
[flaml.automl.logger: 09-23 16:41:19] {2391} INFO -  at 385.9s, estimator lrl1's best error=1.0681,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:41:19] {2218} INFO - iteration 116, current learner lgbm
[flaml.automl.logger: 09-23 16:42:06] {2391} INFO -  at 432.6s, estimator lgbm's best error=0.0899,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:42:06] {2218} INFO - iteration 117, current learner lgbm
[flaml.automl.logger: 09-23 16:43:39] {2391} INFO -  at 525.9s, estimator lgbm's best error=0.0899,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:43:39] {2218} INFO - iteration 118, current learner extra_tree
[flaml.automl.logger: 09-23 16:43:40] {2391} INFO -  at 526.4s, estimator extra_tree's best error=0.7052,       best estimator lgbm'
[flaml.automl.logger: 09-23 16:43:40] {2218} INFO - iteration 119, current learner rf
[flaml.automl.logger: 09-23 16:43:41] {2391} INFO -  at 528.1s, estimator rf's best error=0.5821,       best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:43:41] {2218} INFO - iteration 120, current learner lgbm
[flaml.automl.logger: 09-23 16:43:50] {2391} INFO -  at 536.3s, estimator lgbm's best error=0.0899,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:43:50] {2218} INFO - iteration 121, current learner xgboost
/usr/local/lib/python3.10/dist-packages/xgboost/sklearn.py:1395: UserWarning: `use_label_encoder` is deprecated in 1.7.0.
  warnings.warn("`use_label_encoder` is deprecated in 1.7.0.")
[flaml.automl.logger: 09-23 16:43:52] {2391} INFO -  at 539.0s, estimator xgboost's best error=0.4427,  best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:43:52] {2218} INFO - iteration 122, current learner xgboost
/usr/local/lib/python3.10/dist-packages/xgboost/sklearn.py:1395: UserWarning: `use_label_encoder` is deprecated in 1.7.0.
  warnings.warn("`use_label_encoder` is deprecated in 1.7.0.")
[flaml.automl.logger: 09-23 16:43:55] {2391} INFO -  at 541.9s, estimator xgboost's best error=0.4427,  best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:43:55] {2218} INFO - iteration 123, current learner extra_tree
[flaml.automl.logger: 09-23 16:43:59] {2391} INFO -  at 545.5s, estimator extra_tree's best error=0.6032,       best estimator lgbm'
[flaml.automl.logger: 09-23 16:43:59] {2218} INFO - iteration 124, current learner extra_tree
[flaml.automl.logger: 09-23 16:44:02] {2391} INFO -  at 548.7s, estimator extra_tree's best error=0.6032,       best estimator lgbm'
[flaml.automl.logger: 09-23 16:44:02] {2218} INFO - iteration 125, current learner rf
[flaml.automl.logger: 09-23 16:44:04] {2391} INFO -  at 550.5s, estimator rf's best error=0.5821,       best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:44:04] {2218} INFO - iteration 126, current learner lgbm
[flaml.automl.logger: 09-23 16:45:31] {2391} INFO -  at 637.9s, estimator lgbm's best error=0.0855,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:45:31] {2218} INFO - iteration 127, current learner lgbm
[flaml.automl.logger: 09-23 16:46:10] {2391} INFO -  at 676.3s, estimator lgbm's best error=0.0855,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:46:10] {2218} INFO - iteration 128, current learner lgbm
[flaml.automl.logger: 09-23 16:50:23] {2391} INFO -  at 929.6s, estimator lgbm's best error=0.0855,     best estimator lgbm's best e
[flaml.automl.logger: 09-23 16:51:48] {2627} INFO - retrain lgbm for 85.6s
[flaml.automl.logger: 09-23 16:51:48] {2630} INFO - retrained model: LGBMClassifier(colsample_bytree=0.5082460440659763,
               learning_rate=0.09695015643503321, max_bin=255,
               min_child_samples=4, n_estimators=1, n_jobs=-1, num_leaves=2900,
               reg_alpha=0.0009765625, reg_lambda=0.015515113947203524,
               verbose=-1)
[flaml.automl.logger: 09-23 16:51:48] {1930} INFO - fit succeeded
[flaml.automl.logger: 09-23 16:51:48] {1931} INFO - Time taken to find the best model: 637.8871006965637
```

```
print('Best ML leaner:', automl.best_estimator)
print('Best hyperparmeter config:', automl.best_config)
print('Best log_loss on validation data: {0:.4g}'.format(automl.best_loss))
print('Training duration of best run: {0:.4g} s'.format(automl.best_config_train_time))
```

```
Best ML leaner: lgbm
Best hyperparmeter config: {'n_estimators': 247, 'num_leaves': 2900, 'min_child_samples': 4, 'learning_rate': 0.09695015643503321, 'log_
Best log_loss on validation data: 0.08547
Training duration of best run: 85.56 s
```

```
from sklearn.metrics import classification_report
print(classification_report(y_train, automl.predict(X_train)))
```

```
                  precision    recall  f1-score   support

            draw       1.00      1.00      1.00     26683
            loss       1.00      1.00      1.00     26683
             win       1.00      1.00      1.00     26683

        accuracy                           1.00     80049
       macro avg       1.00      1.00      1.00     80049
    weighted avg       1.00      1.00      1.00     80049
```

```
print(classification_report(y_test, automl.predict(X_test)))
```

```
                  precision    recall  f1-score   support

            draw       0.49      0.33      0.39       968
            loss       0.81      0.79      0.80      2495
             win       0.89      0.94      0.92      6671

        accuracy                           0.85     10134
       macro avg       0.73      0.69      0.70     10134
    weighted avg       0.83      0.85      0.84     10134
```

```
y_pred = automl.predict(test)
y_pred[:5]
```

```
array(['loss', 'win', 'win', 'win', 'win'], dtype=object)
```

```
sol = pd.DataFrame(y_pred,columns=['prediction'])
sol.head()
```

	prediction
0	loss
1	win
2	win
3	win
4	win

```
sol.to_csv('./log_loss.csv',index=False)
```