The Scratch of Machine Learning to Mastering your Data Science Flow

Outlines

1. About Me
2. Types of Machine Learning
    a. Supervised Learning
    b. Unsupervised Learning
    c. Reinforcement Learning
3. Brief Introduction to ML Algorithm
    a. Supervised Learning
    b. Reinforcement Learning
1. References and resources
2. Q&A Section

1. About Me

Siti Khotijah

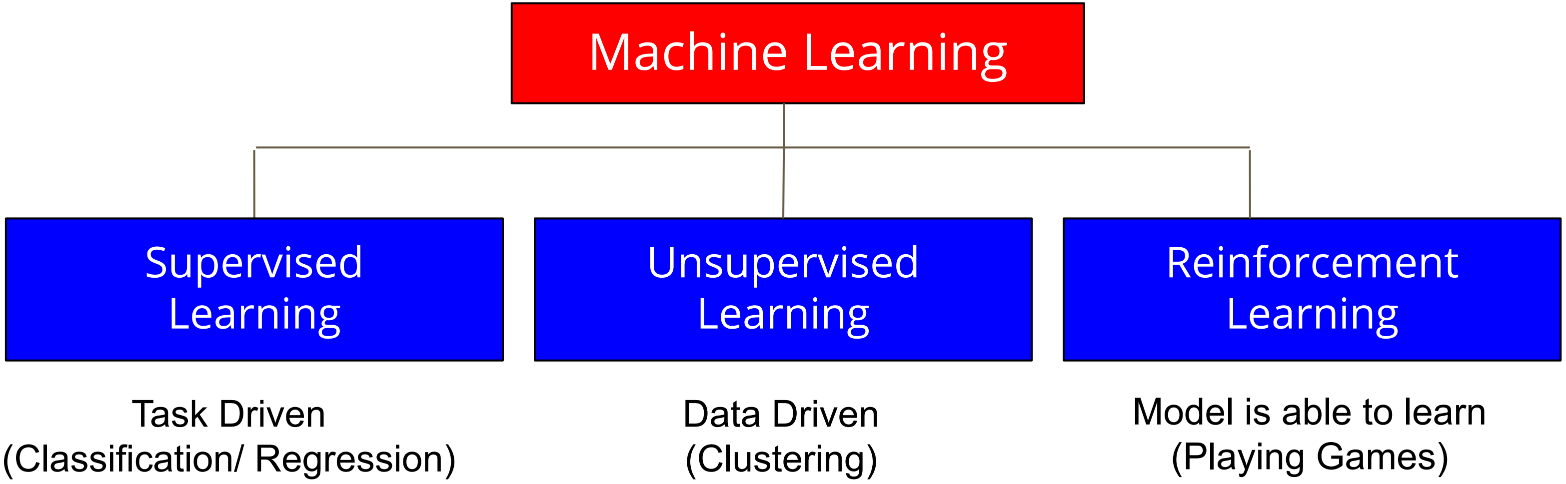
- Masters of Computer Science || Telkom University
- Kaggle Notebooks Master || Competitions Expert

How to reach me:

LinkedIn : Siti Khotijah

Kaggle : Siti Khotijah

2. Types of Machine Learning



a. Supervised Learning

Model make predictions or take decision based on past data

Input raw data

- Please call our customer service representative on 0800 169 6031 between 10am-9pm as you have WON a guaranteed å£1000 cash or å£5000 prize!,,,
- URGENT! Your Mobile No. was awarded å £2000 Bonus Caller Prize on 5/9/03 This is our final try to contact U! Call from Landline 09064019788 BOX42WR29C, 150PPM",,,
- Good stuff, will do.",,,
- I'm leaving my house now...,,,

Processing → Algorithm → Model Trained

Input test

PRIVATE! Your 2004 Account Statement for 07742676969 shows 786 unredeemed Bonus Points. To claim call 08719180248 Identifier Code: 45239 Expires,,,

Model Trained → Output → **It is spam**

The most widely used supervised learning approaches include:

- Linear Regression
- Logistic Regression
- Decision Trees
- Gradient Boosted Trees
- Random Forest
- Support Vector Machines
- K-Nearest Neighbors etc.

b. Unsupervised Learning

Model is able to identify pattern, anomalies, and relationship in the input data

Input raw data

sepal_length,sepal_width,petal_length,petal_width,
5.1,3.5,1.4,0.2
4.9,3,1.4,0.2
6.8,2.8,4.8,1.4
6.7,3,5,1.7
6.3,2.9,5.6,1.8
6.5,3,5.8,2.2

Interpretation → Algorithm → Processing →

Model training

Output

Iris-setosa
Iris-versicolor
Iris-virginica

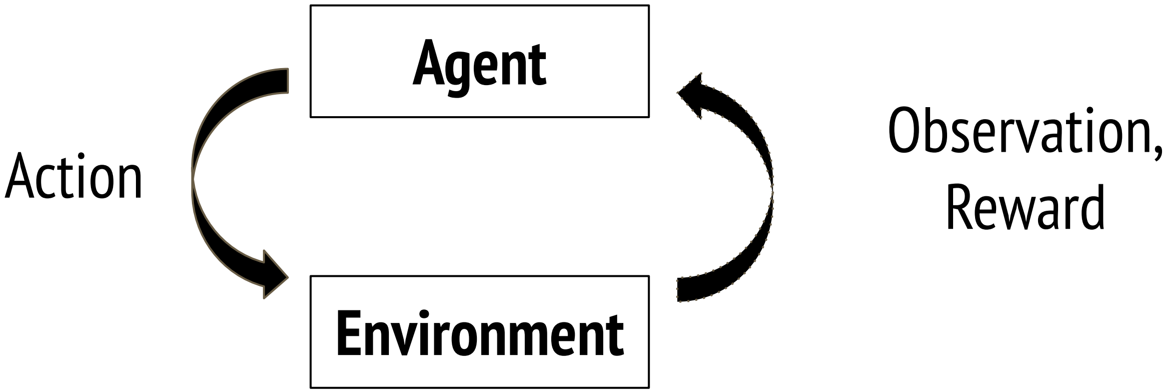
Model trained

Popular techniques used in Unsupervised learning:

- k-means clustering
- t-SNE (t-distributed Stochastic Neighbor Embedding)
- PCA (Principal Component Analysis)
- Association Rule

c. Reinforcement Learning

Model is able to learn based on the rewards it received for it's previous action



Most common reinforcement learning algorithm include:

- Q-Learning
- Temporal Difference (TD)
- Monte-Carlo Tree Search (MCTS)
- Asynchronous Actor-Critic Agents (A3C)

3. Brief Introduction to ML Algorithm

a. Supervised Learning

Tabular Playground Series - Feb 2021

1. Import library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas_profiling import ProfileReport
from lightgbm import LGBMRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold, StratifiedKFold, GroupKFold
from tqdm.notebook import tqdm
from sklearn.preprocessing import LabelEncoder
import datetime
from sklearn.metrics import mean_squared_error, mean_absolute_error
import gc
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

2. Load the data

```
train = pd.read_csv('../input/tabular-playground-series-feb-2021/train.csv')
test = pd.read_csv('../input/tabular-playground-series-feb-2021/test.csv')
sub = pd.read_csv('../input/tabular-playground-series-feb-2021/sample_submission.csv')
```

a. Supervised Learning →

Train data

	id	cat0	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	...	cont5	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	target
0	1	A	B	A	A	B	D	A	E	C	...	0.881122	0.421650	0.741413	0.895799	0.802461	0.724417	0.701915	0.877618	0.719903	6.994023
1	2	B	A	A	A	B	B	A	E	A	...	0.440011	0.346230	0.278495	0.593413	0.546056	0.613252	0.741289	0.326679	0.808464	8.071256
2	3	A	A	A	C	B	D	A	B	C	...	0.914155	0.369602	0.832564	0.865620	0.825251	0.264104	0.695561	0.869133	0.828352	5.760456
3	4	A	A	A	C	B	D	A	E	G	...	0.934138	0.578930	0.407313	0.868099	0.794402	0.494269	0.698125	0.809799	0.614766	7.806457
4	6	A	B	A	A	B	B	A	E	C	...	0.382600	0.705940	0.325193	0.440967	0.462146	0.724447	0.683073	0.343457	0.297743	6.868974
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
299995	499993	A	B	A	C	B	B	A	E	E	...	0.269578	0.258655	0.363598	0.300619	0.340516	0.235711	0.383477	0.215227	0.793630	8.343538
299996	499996	A	B	A	C	B	B	A	E	E	...	0.197211	0.257024	0.574304	0.227035	0.322583	0.286094	0.324874	0.306933	0.230902	7.851861
299997	499997	A	B	A	C	B	B	A	E	C	...	0.449482	0.386172	0.476217	0.135947	0.502730	0.235788	0.316671	0.250286	0.349041	7.600558
299998	499998	A	B	B	C	B	B	A	D	E	...	0.363130	0.324132	0.229017	0.220888	0.515304	0.389391	0.245234	0.303895	0.481138	8.272095
299999	499999	A	A	B	A	B	D	A	E	C	...	0.734712	0.404145	0.497719	0.497974	0.782585	0.751251	0.608412	0.712868	0.452400	6.025685

Test data

	id	cat0	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	...	cont4	cont5	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13
0	0	A	B	A	C	B	D	A	E	E	...	0.701679	0.595507	0.286912	0.279884	0.202234	0.242654	0.285147	0.264308	0.653654	0.302448
1	5	A	B	A	C	B	D	A	E	C	...	0.277480	0.479552	0.397436	0.476742	0.857073	0.516393	0.562065	0.730542	0.318492	0.736251
2	15	A	B	A	C	B	D	A	E	C	...	0.279508	0.676395	0.695284	0.253316	0.586934	0.548555	0.836193	0.759788	0.333572	0.273905
3	16	A	A	B	A	B	D	A	E	E	...	0.479503	0.759875	0.240049	0.298074	0.442475	0.596746	0.414131	0.255382	0.589080	0.311625
4	17	A	B	A	A	B	B	A	E	E	...	0.757845	0.210232	0.329851	0.616663	0.170475	0.263235	0.710961	0.224045	0.285860	0.794931
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
199995	499987	A	A	A	C	B	D	A	E	G	...	0.277365	0.963678	0.240482	0.686462	0.915165	0.848878	0.459598	0.590327	0.864873	0.425258
199996	499990	A	A	A	C	B	D	A	E	E	...	0.523174	0.232072	0.363421	0.694092	0.137002	0.319465	0.364527	0.388908	0.664357	0.224215
199997	499991	A	A	A	C	B	D	A	E	C	...	0.517103	0.432927	0.811876	0.328398	0.496017	0.538779	0.466338	0.643869	0.749590	0.457702
199998	499994	A	B	A	A	B	D	A	E	C	...	0.279153	0.837712	0.680886	0.534439	0.501588	0.809053	0.631704	0.766426	0.937139	0.796304
199999	499995	A	B	A	C	B	C	A	E	G	...	0.763246	0.792263	0.409425	0.285033	0.594721	0.824892	0.479586	0.683065	0.721518	0.722690

a. Supervised Learning →

3. Processing

```
columns = test.columns[1:]
columns
```

```
Index(['cat0', 'cat1', 'cat2', 'cat3', 'cat4', 'cat5', 'cat6', 'cat7', 'cat8',
       'cat9', 'cont0', 'cont1', 'cont2', 'cont3', 'cont4', 'cont5', 'cont6',
       'cont7', 'cont8', 'cont9', 'cont10', 'cont11', 'cont12', 'cont13'],
      dtype='object')
```

```
## Print the categorical columns
cat_features = columns[:10]
cat_features
```

```
Index(['cat0', 'cat1', 'cat2', 'cat3', 'cat4', 'cat5', 'cat6', 'cat7', 'cat8',
       'cat9'],
      dtype='object')
```

```
for feature in cat_features:
    le = LabelEncoder()
    le.fit(train[feature])
    train[feature] = le.transform(train[feature])
    test[feature] = le.transform(test[feature])
```

a. Supervised Learning →

Train data after processing

	id	cat0	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	...	cont5	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	target
0	1	0	1	0	0	1	3	0	4	2	...	0.881122	0.421650	0.741413	0.895799	0.802461	0.724417	0.701915	0.877618	0.719903	6.994023
1	2	1	0	0	0	1	1	0	4	0	...	0.440011	0.346230	0.278495	0.593413	0.546056	0.613252	0.741289	0.326679	0.808464	8.071256
2	3	0	0	0	2	1	3	0	1	2	...	0.914155	0.369602	0.832564	0.865620	0.825251	0.264104	0.695561	0.869133	0.828352	5.760456
3	4	0	0	0	2	1	3	0	4	6	...	0.934138	0.578930	0.407313	0.868099	0.794402	0.494269	0.698125	0.809799	0.614766	7.806457
4	6	0	1	0	0	1	1	0	4	2	...	0.382600	0.705940	0.325193	0.440967	0.462146	0.724447	0.683073	0.343457	0.297743	6.868974

Test data after processing

	id	cat0	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	...	cont4	cont5	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13
0	0	0	1	0	2	1	3	0	4	4	...	0.701679	0.595507	0.286912	0.279884	0.202234	0.242654	0.285147	0.264308	0.653654	0.302448
1	5	0	1	0	2	1	3	0	4	2	...	0.277480	0.479552	0.397436	0.476742	0.857073	0.516393	0.562065	0.730542	0.318492	0.736251
2	15	0	1	0	2	1	3	0	4	2	...	0.279508	0.676395	0.695284	0.253316	0.586934	0.548555	0.836193	0.759788	0.333572	0.273905
3	16	0	0	1	0	1	3	0	4	4	...	0.479503	0.759875	0.240049	0.298074	0.442475	0.596746	0.414131	0.255382	0.589080	0.311625
4	17	0	1	0	0	1	1	0	4	4	...	0.757845	0.210232	0.329851	0.616663	0.170475	0.263235	0.710961	0.224045	0.285860	0.794931

a. Supervised Learning →

4. Training

```python
target = train['target'].values
```

```python
train_oof = np.zeros((300000,))
test_preds = 0
train_oof.shape
```

```
(300000,)
```

```python
params = {'max_depth': 16,
          'subsample': 0.8032697250789377,
          'colsample_bytree': 0.21067140508531404,
          'learning_rate': 0.009867383057779643,
          'reg_lambda': 10.987474846877767,
          'reg_alpha': 17.335285595031994,
          'min_child_samples': 31,
          'num_leaves': 66,
          'max_bin': 522,
          'cat_smooth': 81,
          'cat_l2': 0.029690334194270022,
          'metric': 'rmse',
          'n_jobs': -1,
          'n_estimators': 30000}
```

a. Supervised Learning →

LGBM Regressor

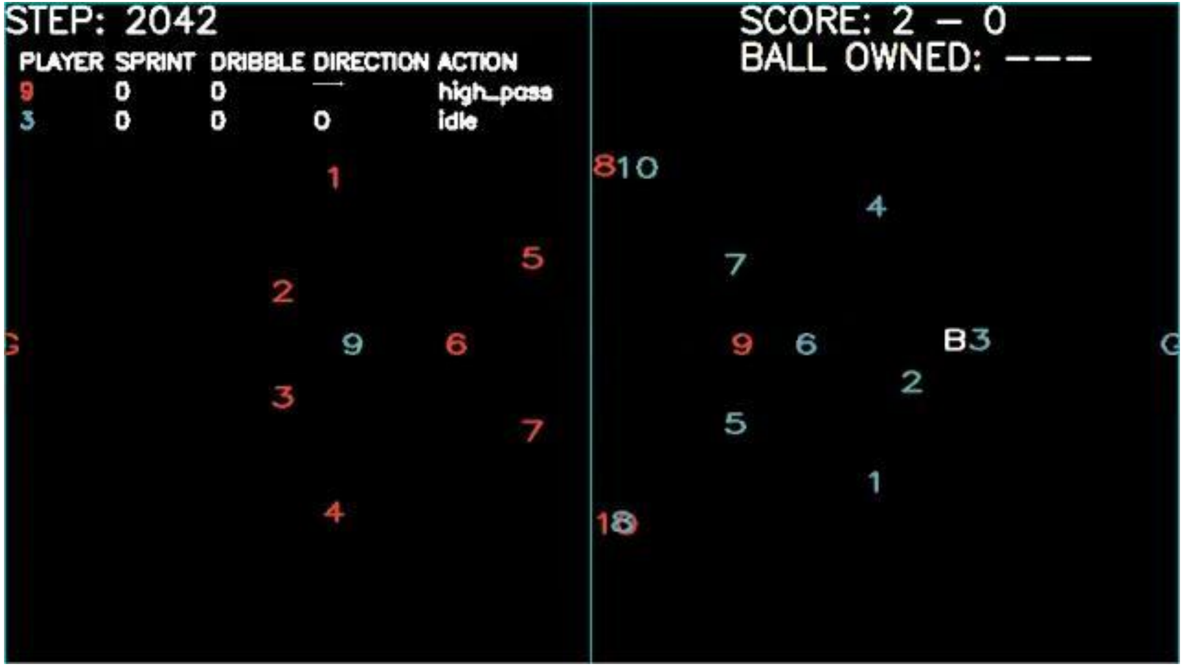
```
NUM_FOLDS = 10
kf = KFold(n_splits=NUM_FOLDS, shuffle=True, random_state=2021)

for f, (train_ind, val_ind) in tqdm(enumerate(kf.split(train, target))):
        #print(f'Fold {f}')
        train_df, val_df = train.iloc[train_ind][columns], train.iloc[val_ind][columns]
        train_target, val_target = target[train_ind], target[val_ind]

        model = LGBMRegressor(**params)
        model.fit(train_df, train_target, eval_set=[(val_df,val_target)],early_stopping_
rounds=2000,verbose=False)
        temp_oof = model.predict(val_df)
        temp_test = model.predict(test[columns])

        train_oof[val_ind] = temp_oof
        test_preds += temp_test/NUM_FOLDS

        print(mean_squared_error(temp_oof, val_target, squared=False))
```

```
mean_squared_error(train_oof, target, squared=False)
```

```
sub['target'] = test_preds
sub.to_csv('submission.csv', index=False)
```

→ **See details**

b. Reinforcement Learning



→ See details

4. References and resources

1. Machine Learning Youtube Playlist
2. SMS Spam Collection Dataset
3. Iris Flower Dataset
4. Tabular Playground Series - Feb 2021
5. Google Research Football with Manchester City F.C.

5. Q&A section