Machine Learning

Pembelajaran mesin adalah cabang kecerdasan buatan (AI) yang difokuskan pada pembangunan aplikasi yang belajar dari data dan meningkatkan ketepatannya dari masa ke masa tanpa diprogramkan untuk melakukannya.

Supervised machine learning

Pembelajaran mesin yang diselia melatih dirinya pada set data berlabel. Artinya, data dilabel dengan informasi yang model pembelajaran mesin sedang dibangun untuk menentukan dan bahkan mungkin diklasifikasikan dengan cara model tersebut seharusnya mengklasifikasikan data. Sebagai contoh, model penglihatan komputer yang direka untuk mengenal pasti anjing Gembala Jerman yang tulen mungkin dilatih menggunakan kumpulan data pelbagai gambar anjing berlabel.

Unsupervised machine learning

Pembelajaran mesin yang tidak diawasi memakan data yang tidak berlabel — banyak dan banyak — dan menggunakan algoritma untuk mengekstrak ciri-ciri bermakna yang diperlukan untuk melabel, menyusun dan mengklasifikasikan data dalam masa nyata, tanpa campur tangan manusia. Pembelajaran tanpa pengawasan lebih kurang mengotomatisasi keputusan dan ramalan, dan lebih banyak mengenai mengenal pasti corak dan hubungan dalam data yang akan dilewatkan oleh manusia. Sebagai contoh, ikuti pengesanan spam - orang menghasilkan lebih banyak e-mel daripada yang mungkin diharapkan oleh pasukan saintis data untuk melabel atau mengklasifikasikannya sepanjang hayat mereka. Algoritma pembelajaran yang tidak diawasi dapat menganalisis sejumlah besar e-mel dan mengungkap ciri dan corak yang menunjukkan spam (dan terus menjadi lebih baik dalam menandai spam dari masa ke masa).

Contoh kes penggunaan machine learning

- **Pembantu digital: Apple Siri, Amazon Alexa, Pembantu Google, dan pembantu digital lain dikuasakan oleh pemprosesan bahasa semula jadi (NLP), aplikasi pembelajaran mesin yang membolehkan komputer memproses data teks dan suara dan 'memahami' bahasa manusia seperti yang dilakukan orang. Pemprosesan bahasa semula jadi juga mendorong aplikasi yang didorong oleh suara seperti perisian GPS dan pengecaman pertuturan (ucapan-ke-teks).**
- **Cadangan: Model pembelajaran mendalam mendorong cadangan 'orang juga disukai' dan 'hanya untuk anda' yang ditawarkan oleh Amazon, Netflix, Spotify, dan perkhidmatan runcit, hiburan, perjalanan, pencarian pekerjaan, dan berita lain.**
- **Pengiklanan dalam talian kontekstual: Model pembelajaran mesin dan pembelajaran mendalam dapat menilai kandungan halaman web - tidak hanya topik, tetapi nuansa seperti pendapat atau sikap pengarang - dan menayangkan iklan yang disesuaikan dengan minat pengunjung.**
- **Chatbots: Chatbots dapat menggunakan kombinasi pengecaman corak, pemprosesan bahasa semula jadi, dan rangkaian neural mendalam untuk menafsirkan teks input dan memberikan respons yang sesuai.**
- **Pengesanan penipuan: Model regresi dan klasifikasi pembelajaran mesin telah menggantikan sistem pengesanan penipuan berdasarkan peraturan, yang mempunyai jumlah positif yang tinggi ketika menandakan penggunaan kad kredit yang dicuri dan jarang berjaya mengesan penggunaan data kewangan yang dicuri atau dikompromikan secara jenayah.**
- **Keselamatan siber: Pembelajaran mesin dapat mengekstrak kecerdasan dari laporan kejadian, amaran, catatan blog, dan banyak lagi untuk mengenal pasti potensi ancaman, memberi nasihat kepada penganalisis keselamatan, dan mempercepat tindak balas.**
- **Analisis imej perubatan: Jenis dan jumlah data pengimejan perubatan digital telah meletup, membawa kepada lebih banyak maklumat yang tersedia untuk menyokong diagnosis tetapi juga lebih banyak peluang untuk kesilapan manusia dalam membaca data. Rangkaian neural konvolusional (CNN), rangkaian neural berulang (RNN), dan model pembelajaran mendalam lain telah terbukti semakin berjaya dalam mengekstrak ciri dan maklumat dari gambar perubatan untuk membantu menyokong diagnosis yang tepat.**
- **Kereta memandu sendiri: Kereta memandu sendiri memerlukan latihan pembelajaran mesin secara automatik — mereka mesti terus mengenal pasti objek di persekitaran di sekitar kereta, meramalkan bagaimana ia akan berubah atau bergerak, dan memandu kereta di sekitar objek dan juga ke arah pemandu destinasi. Hampir setiap bentuk pembelajaran mesin dan algoritma pembelajaran mendalam yang**

disebutkan di atas memainkan beberapa peranan dalam mengaktifkan kenderaan memandu sendiri.

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
data = pd.read_excel("data.xlsx", sheet_name=5)
```

In [3]:

```
data.head()
```

Out[3]:

	id	loc_id	veh_type	avg_speed	ttl_count	over_speed	created_at	Speeding Percentage Vehicle (%)	Total Non-speeding vehicle
0	166b0980daf0d97e	2	Lorry	0	0	0	2021-03-11 00:55:40	NaN	0
1	166b0701380741c0	2	Lorry	0	0	0	2021-03-11 00:09:53	NaN	0
2	166b0b23e562095c	2	Lorry	0	0	0	2021-03-11 01:25:40	NaN	0
3	166b0bf577701a4e	2	Lorry	0	0	0	2021-03-11 01:40:40	NaN	0
4	166b0cc7051bc53e	2	Lorry	0	0	0	2021-03-11 01:55:40	NaN	0

In [4]:

```
data.fillna(0, inplace=True)
```

In [5]:

```
data['created_at'] = pd.to_datetime(data['created_at'])
data['hour'] = data['created_at'].dt.hour
```

In [6]:

```
veh_type_dum = pd.get_dummies(data['veh_type'])
data = pd.concat([data, veh_type_dum], axis=1)
```

In [7]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1413 entries, 0 to 1412
Data columns (total 13 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   id                               1413 non-null   object
 1   loc_id                           1413 non-null   int64
 2   veh_type                         1413 non-null   object
 3   avg_speed                        1413 non-null   int64
 4   ttl_count                        1413 non-null   int64
 5   over_speed                       1413 non-null   int64
 6   created_at                       1413 non-null   datetime64[ns]
 7   Speeding Percentage Vehicle (%)  1413 non-null   float64
 8   Total Non-speeding vehicle       1413 non-null   int64
 9   hour                             1413 non-null   int64
 10  Car                              1413 non-null   uint8
 11  Lorry                            1413 non-null   uint8
 12  Motorcycle                       1413 non-null   uint8
```

```
dtypes: datetime64[ns](1), float64(1), int64(6), object(2), uint8(3)
memory usage: 114.7+ KB
```

In [8]:

```
data.nunique()
```

Out[8]:

```
id                                1413
loc_id                               1
veh_type                             3
avg_speed                           55
ttl_count                          170
over_speed                          22
created_at                         476
Speeding Percentage Vehicle (%)    182
Total Non-speeding vehicle         169
hour                                24
Car                                  2
Lorry                                2
Motorcycle                           2
dtype: int64
```

Jenis-jenis machine learning

Classification

Dalam pembelajaran mesin, klasifikasi merujuk kepada masalah pemodelan ramalan di mana label kelas diramalkan untuk contoh data input yang diberikan.

Contoh masalah pengelasan termasuk:

- **Diberi contoh, klasifikasikan sama ada spam atau tidak.**
- **Memandangkan watak tulisan tangan, klasifikasikannya sebagai salah satu watak yang diketahui.**
- **Memandangkan tingkah laku pengguna baru-baru ini, klasifikasikan sebagai perubahan atau tidak.**

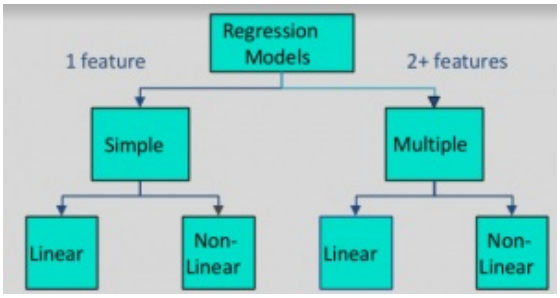
Dari perspektif pemodelan, klasifikasi memerlukan set data latihan dengan banyak contoh input dan output yang dapat dipelajari.

Jenis-jenis classification:

1. **Binary Classification (spam or not)**
2. **Multi-Class Classification (Face classification)**
3. **Multi-Label Classification (Image has apple or banana or pear)**
4. **Imbalanced Classification (Fraud detection)**

Regression

Masalah regresi adalah apabila pemboleh ubah output adalah nilai nyata atau berterusan, seperti "gaji" atau "berat". Banyak model yang berbeza dapat digunakan, yang paling mudah adalah regresi linier. Ia cuba menyesuaikan data dengan hiper-pesawat terbaik yang melalui titik.



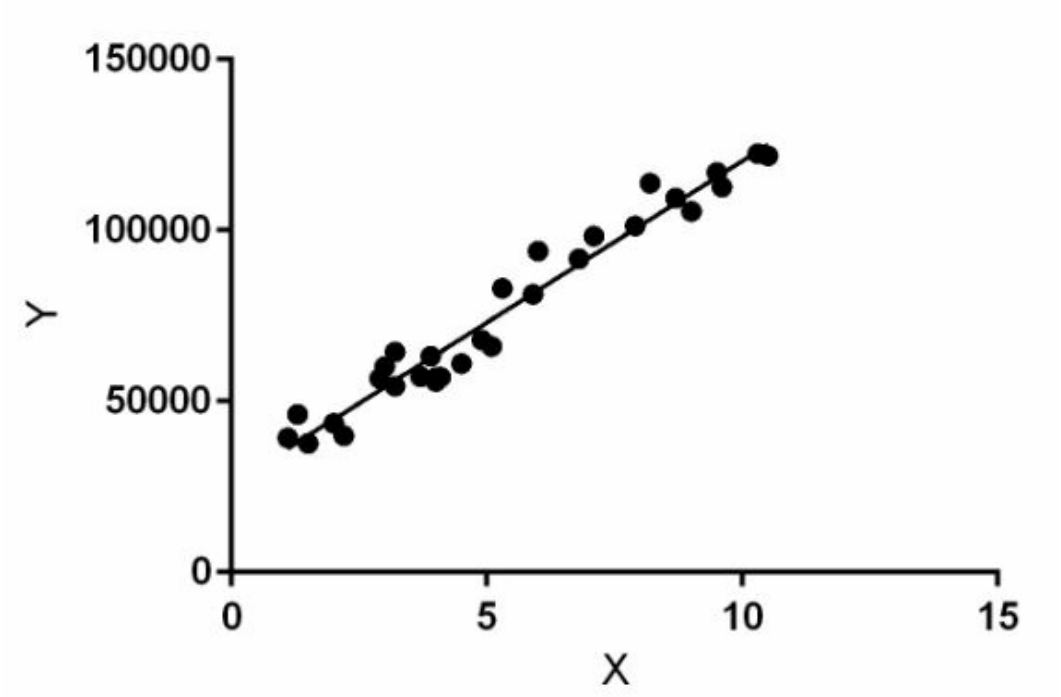
In [9]:

```
# Dalam kes yang hendak diselesaikan, kita harus menggunakan regression models, pada baha
```

```
gian 2 nanti kite akan menggunakan model classification
from sklearn.linear_model import LinearRegression # Linear Model
from sklearn.ensemble import RandomForestRegressor # Bagging Model
import lightgbm as lgb # Boosting Model
```

Linear Regression

Linear Regression adalah algoritma pembelajaran mesin berdasarkan pembelajaran yang diselia. Ia melakukan tugas regresi. Regresi memodelkan nilai ramalan sasaran berdasarkan pemboleh ubah bebas. Ia digunakan untuk mengetahui hubungan antara pemboleh ubah dan ramalan. Model regresi yang berbeza berbeza berdasarkan - jenis hubungan antara pemboleh ubah bersandar dan bebas, mereka mempertimbangkan dan jumlah pemboleh ubah bebas yang digunakan.



In [10]:

```
# Dengan menggunakan perpustakaan sklearn, kita dapat menggunakan semua model pembelajara
n mesin yang disediakan dalam AI moden
# Cara untuk menyatakan model jenis Linear Regression
lr = LinearRegression() # Tidak ada hyperparameters

# Cara untuk menyatakan pemboleh ubah untuk latihan pembelajaran mesin
X = data[['avg_speed', 'ttl_count', 'over_speed', 'hour', 'Car', 'Lorry', 'Motorcycle']]
.values
y = data['Speeding Percentage Vehicle (%)']

print(X.shape, y.shape)

print(X)
```

```
(1413, 7) (1413,)
[[  0   0   0 ...   0   1   0]
 [  0   0   0 ...   0   1   0]
 [  0   0   0 ...   0   1   0]
 ...
 [ 48 173  12 ...   1   0   0]
 [ 48 125   6 ...   1   0   0]
 [ 48  87  10 ...   1   0   0]]
```

In [11]:

```
# Model latihan pembelajaran mesin adalah proses di mana algoritma pembelajaran mesin (ML
) diberi data latihan yang mencukupi untuk dipelajari.
# Kaedah untuk memulakan latihan
lr.fit(X, y)
```

Out[11]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [12]:

```
# Ramalan merujuk pada output algoritma setelah dilatih pada set data historis dan ditera
pkan pada data baru ketika meramalkan kemungkinan
# hasil tertentu, seperti apakah pelanggan akan berpindah dalam 30 hari atau tidak.
# Algoritma akan menghasilkan kemungkinan nilai untuk pemboleh ubah yang tidak diketahui
untuk setiap rekod dalam data baru,
# yang membolehkan pembangun model mengenal pasti nilai yang kemungkinan besar.

# Kaedah untuk mendapatkan ramalan daripada model yang sudah dilatih

preds = lr.predict(X)

# Kita akan menambahkan ramalan ke kerangka data utama untuk tujuan penilaian
data['Prediction Speeding Pct'] = preds

data.head()
```

Out[12]:

	id	loc_id	veh_type	avg_speed	ttl_count	over_speed	created_at	Speeding Percentage Vehicle (%)	Total Non-speeding vehicle	hour	Car	Lorry
0	166b0980daf0d97e	2	Lorry	0	0	0	2021-03-11 00:55:40	0.0	0	0	0	1
1	166b0701380741c0	2	Lorry	0	0	0	2021-03-11 00:09:53	0.0	0	0	0	1
2	166b0b23e562095c	2	Lorry	0	0	0	2021-03-11 01:25:40	0.0	0	1	0	1
3	166b0bf577701a4e	2	Lorry	0	0	0	2021-03-11 01:40:40	0.0	0	1	0	1
4	166b0cc7051bc53e	2	Lorry	0	0	0	2021-03-11 01:55:40	0.0	0	1	0	1

Evaluation

Model Evaluasi adalah bahagian yang tidak terpisahkan dari proses pengembangan model. Ia membantu mencari model terbaik yang mewakili data kami dan seberapa baik model yang dipilih akan berfungsi pada masa hadapan. Menilai prestasi model dengan data yang digunakan untuk latihan tidak dapat diterima dalam sains data kerana dapat dengan mudah menghasilkan model yang terlalu optimis dan berlebihan. Terdapat dua kaedah menilai model dalam sains data, Hold-Out dan Cross-Validation. Untuk mengelakkan overfitting, kedua-dua kaedah menggunakan set ujian (tidak dilihat oleh model) untuk menilai prestasi model.

Regression Evaluation

Beberapa jenis evaluasi bagi model regression

1. **Root Mean Squared Error - RMSE adalah formula yang popular untuk mengukur kadar ralat model regresi. Namun, hanya dapat dibandingkan antara model yang kesalahannya diukur dalam unit yang sama.**
2. **Relative Squared Error - Tidak seperti RMSE, ralat kuasa dua relatif (RSE) dapat dibandingkan antara model yang kesalahannya diukur dalam unit yang berlainan.**
3. **Mean Absolute Error - Kesalahan mutlak min (MAE) mempunyai unit yang sama dengan data asal, dan hanya dapat dibandingkan antara model yang kesalahannya diukur dalam unit yang sama. Ia biasanya serupa dengan ukuran RMSE, tetapi sedikit lebih kecil.**

4. **R2 Score - R2 menerangkan bahagian varian pemboleh ubah bersandar yang dijelaskan oleh model regresi. Jika model regresi "sempurna", SSE adalah nol, dan R2 adalah 1. Jika model regresi adalah kegagalan total, SSE sama dengan SST, tidak ada variasi yang dijelaskan oleh regresi, dan R2 adalah nol.**

Classification Evaluation

Jenis evaluasi bagi model classification

1. **Confusion Matrix - Matriks kekeliruan menunjukkan bilangan ramalan yang betul dan salah yang dibuat oleh model klasifikasi berbanding dengan hasil sebenar (nilai sasaran) dalam data. Matriks adalah NxN, di mana N adalah bilangan nilai sasaran (kelas). Prestasi model sedemikian biasanya dinilai menggunakan data dalam matriks. Jadual berikut memaparkan matriks kekeliruan 2x2 untuk dua kelas (Positif dan Negatif).**
   A. **(Accuracy) Ketepatan: bahagian jumlah ramalan yang betul.**
   B. **(Precision) Nilai Ramalan Positif atau Ketepatan: bahagian kes positif yang dikenal pasti dengan betul.**
   C. **(Negative Predictive Value) Nilai Ramalan Negatif: bahagian kes negatif yang dikenal pasti dengan betul.**
   D. **(Recall) Sensitiviti atau Ingat semula: bahagian kes positif sebenar yang dikenal pasti dengan betul.**
   E. **(Specificity)Kekhususan: bahagian kes negatif sebenar yang dikenal pasti dengan betul.**

In [13]:

```
# Mengimport modul-modul bagi evaluasi model regression yang telah dilatih
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from math import sqrt
```

In [14]:

```
# Masa untuk dapatkan skor untuk setiap jenis evaluasi
rmse = sqrt(mean_squared_error(y, preds))
mae = mean_absolute_error(y, preds)
r2 = r2_score(y, preds)
print("RMSE is {}".format(rmse))
print("MAE is {}".format(mae))
print("R2 is {}".format(r2))
```

```
RMSE is 0.13179354305194768
MAE is 0.07596309277170578
R2 is 0.3919172045939412
```

In [15]:

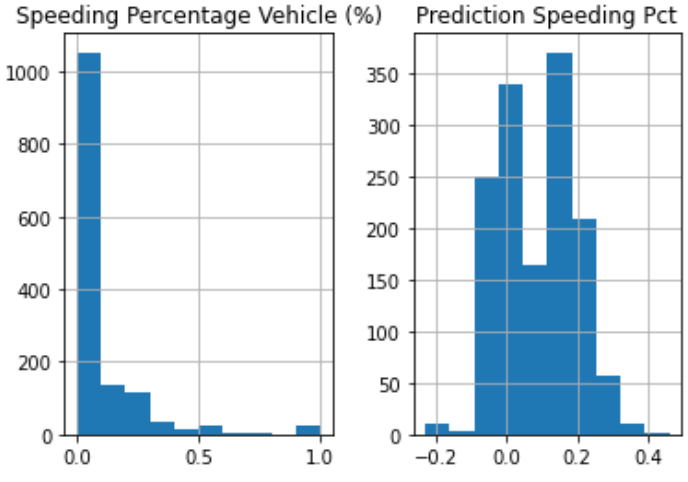
```
import seaborn as sns
```

In [16]:

```
data[['Speeding Percentage Vehicle (%)' ,'Prediction Speeding Pct']].hist()
```

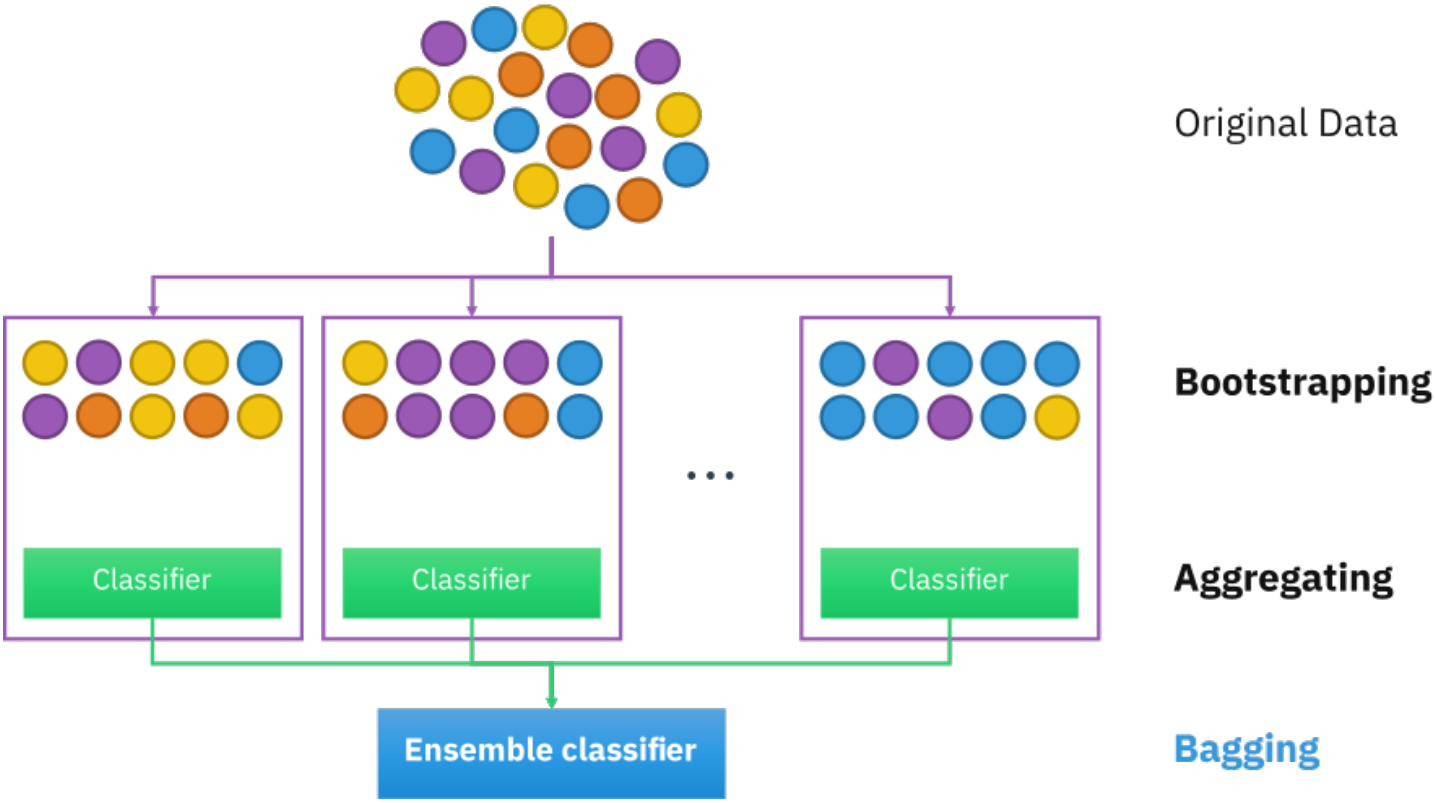
Out[16]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fbc07a1ded0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fbc072faed0>]],
      dtype=object)
```



Bagging Models

Pengumpulan bootstrap, juga disebut bagging (dari agregasi bootstrap), adalah meta-algoritma ensemble pembelajaran mesin yang direka untuk meningkatkan kestabilan dan ketepatan algoritma pembelajaran mesin yang digunakan dalam klasifikasi statistik dan regresi. Ia juga mengurangkan perbezaan dan membantu mengelakkan pemakaian berlebihan. Walaupun biasanya diterapkan pada kaedah pohon keputusan, ia dapat digunakan dengan jenis kaedah apa pun. Bagging adalah kes khas dari pendekatan rata-rata model.



Random Forest Regressor

Random Forest adalah penambahbaikan daripada Bagging.

Masalah dengan Decision Trees seperti CART ialah mereka tamak. Mereka memilih pemboleh ubah mana yang akan dibahagi menggunakan algoritma tamak yang meminimumkan ralat. Oleh itu, walaupun dengan Bagging, Decision Trees dapat memiliki banyak persamaan struktur dan pada gilirannya mempunyai korelasi yang tinggi dalam ramalannya.

Menggabungkan ramalan dari pelbagai model dalam ensemble berfungsi lebih baik jika ramalan dari sub-model tidak berkaitan atau paling tidak lemah.

Hyperparameters yang utama di dalam Random Forest:

1. **n_estimators, default=100 - The number of trees in the forest.**
2. **criterion {“mse”, “mae”}, default=”mse” The function to measure the quality of a split. Supported criteria are “mse” for the mean squared error, which is equal to variance reduction as feature selection criterion, and “mae” for the mean absolute error**
3. **max_depth, default=None The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.**
4. **min_samples_split, default=2 The minimum number of samples required to split an internal node**
5. **min_samples_leaf, default=1 The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression**

In [17]:

```
# Cara untuk menyatakan model Random Forest Regressor
rfr = RandomForestRegressor() # Kita gunakan default hyperparameters dahulu
```

```
# Cara untuk melatih model Random Forest Regressor
rfr.fit(X, y)

# Cara untuk mendapatkan ramalan
preds = rfr.predict(X)

# Menambahkan ramalan baru ke kerangka data utama untuk tujuan penilaian
data['Prediction Speeding Pct'] = preds

data.head()
```

Out[17]:

	id	loc_id	veh_type	avg_speed	ttl_count	over_speed	created_at	Speeding Percentage Vehicle (%)	Total Non-speeding vehicle	hour	Car	Lorry
0	166b0980daf0d97e	2	Lorry	0	0	0	2021-03-11 00:55:40	0.0	0	0	0	1
1	166b0701380741c0	2	Lorry	0	0	0	2021-03-11 00:09:53	0.0	0	0	0	1
2	166b0b23e562095c	2	Lorry	0	0	0	2021-03-11 01:25:40	0.0	0	1	0	1
3	166b0bf577701a4e	2	Lorry	0	0	0	2021-03-11 01:40:40	0.0	0	1	0	1
4	166b0cc7051bc53e	2	Lorry	0	0	0	2021-03-11 01:55:40	0.0	0	1	0	1

In [18]:

```
# Masa untuk dapatkan skor untuk setiap jenis evaluasi
rmse = sqrt(mean_squared_error(y, preds))
mae = mean_absolute_error(y, preds)
r2 = r2_score(y, preds)
print("RMSE is {}".format(rmse))
print("MAE is {}".format(mae))
print("R2 is {}".format(r2))
```

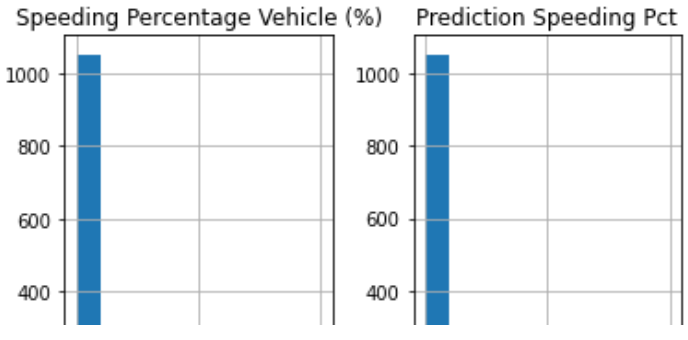
```
RMSE is 0.01103262061023399
MAE is 0.003267315633332611
R2 is 0.9957388009945737
```

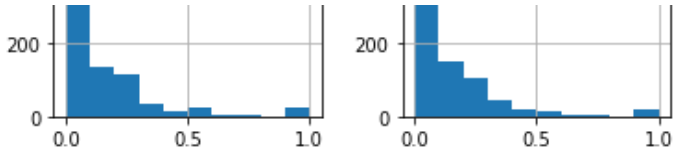
In [19]:

```
data[['Speeding Percentage Vehicle (%)' ,'Prediction Speeding Pct']].hist()
```

Out[19]:

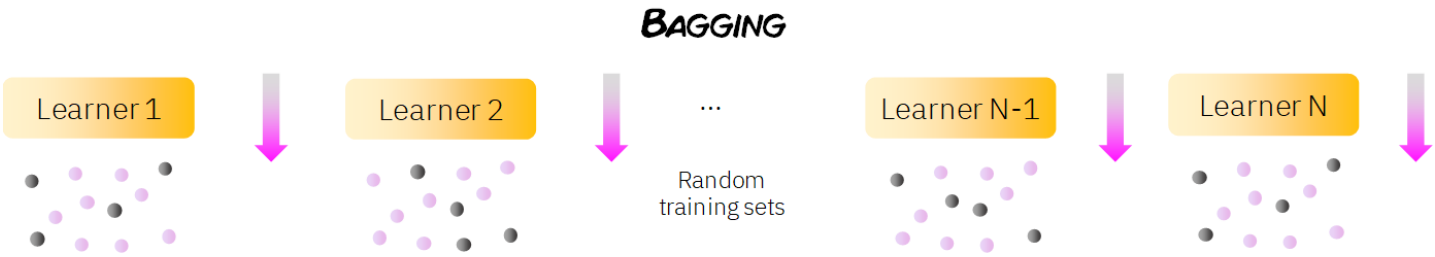
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fbc07734790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fbc06f238d0>]],
      dtype=object)
```





Boosting Models

Boosting, yang awalnya bernama Hypothesis Boosting, terdiri dari idea menyaring atau menimbang data yang digunakan untuk melatih pasukan kami yang lemah, sehingga setiap pelajar baru memberikan lebih banyak berat atau hanya dilatih dengan pemerhatian yang telah diklasifikasikan dengan buruk oleh yang sebelumnya pelajar



HyperParameters in LightGBM

1. **boosting_type (string, optional (default='gbdt')) – 'gbdt', traditional Gradient Boosting Decision Tree. 'dart', Dropouts meet Multiple Additive Regression Trees. 'goss', Gradient-based One-Side Sampling. 'rf', Random Forest.**
2. **num_leaves (int, optional (default=31)) – Maximum tree leaves for base learners.**
3. **max_depth (int, optional (default=-1)) – Maximum tree depth for base learners, <=0 means no limit.**
4. **learning_rate (float, optional (default=0.1)) – Boosting learning rate. You can use callbacks parameter of fit method to shrink/adapt learning rate in training using reset_parameter callback. Note, that this will ignore the learning_rate argument in training.**
5. **n_estimators (int, optional (default=100)) – Number of boosted trees to fit.**
6. **objective (string, callable or None, optional (default=None)) – Specify the learning task and the corresponding learning objective or a custom objective function to be used (see note below). Default: 'regression' for LGBMRegressor, 'binary' or 'multiclass' for LGBMClassifier, 'lambdarank' for LGBMRanker.**

[To find more about lightgbm]

In [20]:

```
# Cara untuk menyatakan model Light Gradient Boosting Model
param = {'num_leaves': 31, 'objective': 'regression', 'n_estimators': 500} # Perlu dinya
takan hyperparameters yang ingin diubah
param['metric'] = 'rmse'

train_data = lgb.Dataset(X, label=y)

# Cara untuk melatih model Light Gradient Boosting Model
num_round = 100
model = lgb.train(param, train_data, num_round)

# Cara untuk mendapatkan ramalan
```

```
preds = model.predict(X)

# Menambahkan ramalan baru ke kerangka data utama untuk tujuan penilaian
data['Prediction Speeding Pct'] = preds

data.head()
```

```
/usr/local/lib/python3.7/dist-packages/lightgbm/engine.py:118: UserWarning: Found `n_esti
mators` in params. Will use it instead of argument
  warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))
```

Out[20]:

	id	loc_id	veh_type	avg_speed	ttl_count	over_speed	created_at	Speeding Percentage Vehicle (%)	Total Non-speeding vehicle	hour	Car	Lorry
0	166b0980daf0d97e	2	Lorry	0	0	0	2021-03-11 00:55:40	0.0	0	0	0	1
1	166b0701380741c0	2	Lorry	0	0	0	2021-03-11 00:09:53	0.0	0	0	0	1
2	166b0b23e562095c	2	Lorry	0	0	0	2021-03-11 01:25:40	0.0	0	1	0	1
3	166b0bf577701a4e	2	Lorry	0	0	0	2021-03-11 01:40:40	0.0	0	1	0	1
4	166b0cc7051bc53e	2	Lorry	0	0	0	2021-03-11 01:55:40	0.0	0	1	0	1

In [21]:

```
# Masa untuk dapatkan skor untuk setiap jenis evaluasi
rmse = sqrt(mean_squared_error(y, preds))
mae = mean_absolute_error(y, preds)
r2 = r2_score(y, preds)
print("RMSE is {}".format(rmse))
print("MAE is {}".format(mae))
print("R2 is {}".format(r2))
```

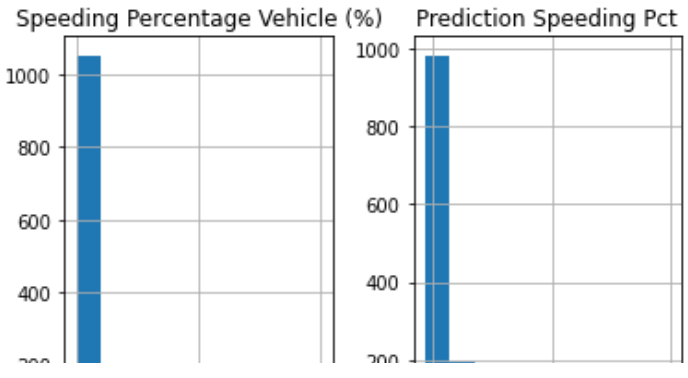
```
RMSE is 0.010799106393043063
MAE is 0.002752060042780309
R2 is 0.9959172753689676
```

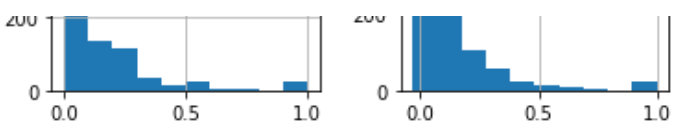
In [22]:

```
data[['Speeding Percentage Vehicle (%)' ,'Prediction Speeding Pct']].hist()
```

Out[22]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fbc077e6a50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fbc077e9c90>]],
      dtype=object)
```





Kesimpulan

Boosting dan Bagging Model telah berjaya untuk menghasilkan ramalan yang lebih tepat jika dibandingkan dengan basci linear regression model. Hal ini kerana, bagging dan boosting model mempunyai kebolehan untuk menyesuaikan latihan terhadap data yang diramal salah pada kitaran latihan pertama, dan ianya akan cuba untuk menyesuaikan algoritma terhadap data yang diramal jauh daripada benar dengan meletakkan 'weight' lebih.

In [ ]: