PRE TEST!

Pre Test no. 1

How to do this Exercise

Option
A. df.drop_duplicates()
B. df.dropduplicates()
C. df.drop_duplicate()
D. df.Total()

How to do this Exercise value is Number

A. df.drop_duplicates()

Pre Test no. 2

How to change integer column to string in Pandas?

Option
A. df[[Column1,Column2,Column3]].astype('int')
B. df[[Column1,Column2,Column3]].astype('str')
C. df[[Column1,Column2,Column3]].astype('float')

A. df[[Column1,Column2,Column3]].astype('int')

Pretest no.3

How to do this Exercise

Pretest no.4

How to change date + "2019-01-01" to datetime format?

Option
A. datetime.strptime(date, %Y-%m-%d')
B. datetime.strftime(date, %Y-%m-%d')
C. datetime.date(date, %Y-%m-%d')
D. datetime.date(date, %Y-%m-%d')

A. datetime.strptime(date, %Y-%m-%d')

How to apply by function into row in pandas?

Option
A. df.apply(lambda x)
B. df.apply(lambda x)
C. df.apply(lambda x)
D. df.apply(lambda x)

Pretest no.5

B. df.applymap(lambda x)

Introduction to Machine Learning

What's on your mind about Machine Learning?

Belajar Machine Learning dan Deep Learning saya dapat dipelajari di Coursera, Udacity dan Pluralsight. Pada Platform

Machine Learning

Machine learning merupakan salah satu cabang dari ilmu Artificial Intelligence. Sementara itu, Deep Learning merupakan bagian machine learning yang lebih spesifik dan lebih powerful dibandingkan dengan machine learning.

Manfaat dari Machine Learning

- Pengolahan Bahasa Alami (Natural Language Processing) Contoh implementasi nyata pada, mulai ada voice dari kata kuasi dari Microsoft Kini lebih kasar dan alami
- Pengenalan Pola (Pattern Recognition)
- Pengenalan Wajah (Face Recognition) Misalnya dimanfaatkan dalam kamera machine learning mengenali bentuk/ciri dari gambarnya. Sehingga bisa fokus ke wajah, bisa juga untuk mengenali wajah seseorang untuk fitur keamanan pada smartphone. Misalnya pada kasus lain, FB dan lain-lain sudah bisa mengenali wajah dan mengenali siapa orang yang ada di foto tersebut
- Pengenalan Gerakan (Gesture Recognition) Misalnya bisa dimanfaatkan untuk kebutuhan game dan gaming. Kinect Dari Jaya saya memberikan fitur kinect ada di game Xbox dan Kinect juga bisa dipakai untuk mengenali gerakan tangan dan kaki. Dengan kamera dan machine learning kita bisa ajak orang untuk berinteraksi dengan game. Sehingga machine learning mampu untuk mengenali gerakan tangan dan kaki. Dengan kamera dan machine learning kita bisa ajak orang untuk berinteraksi dengan game
- Pengolahan Data Besar (Big Data Processing)
- Pengolahan Keputusan (Decision Processing)

Different case need to handle with different type of ML task

Machine Learning Algorithm

Semi Supervised Learning

Tentang kaya akan supervised learning atau kita mendalami pembelajaran dari supervised dan unsupervised. Sebelumnya kita punya 100 data. 100 data tersebutnya kayak dan cukup sebagainya. 100 data tersebut kayak kita punya kayak contoh-contoh kita. Lengkap dengan data-nya. Dengan data-nya kita bisa belajar dengan supervised learning. Nah, kita punya juga data yang tidak lengkap. Misalnya kita punya 1 juta data. Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning.

- Supervised Learning: Tentunya dan machine. Sebelumnya dan supervised. Machine learning dan supervised. Sebelumnya kita punya 100 data. 100 data tersebutnya kayak dan cukup sebagainya. 100 data tersebut kayak kita punya kayak contoh-contoh kita. Lengkap dengan data-nya. Dengan data-nya kita bisa belajar dengan supervised learning. Nah, kita punya juga data yang tidak lengkap. Misalnya kita punya 1 juta data. Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning.
- Unsupervised: Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning. Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning.
- Semi-supervised learning: Tentunya dan machine. Sebelumnya dan supervised. Machine learning dan supervised. Sebelumnya kita punya 100 data. 100 data tersebutnya kayak dan cukup sebagainya. 100 data tersebut kayak kita punya kayak contoh-contoh kita. Lengkap dengan data-nya. Dengan data-nya kita bisa belajar dengan supervised learning. Nah, kita punya juga data yang tidak lengkap. Misalnya kita punya 1 juta data. Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning.

General Step

Langkah:

1. Kumpulkan data
2. Persiapkan data
3. Pilih fitur
4. Pilih algoritma
5. Latih model
6. Evaluasi model
7. Tuning parameter
8. Gunakan model untuk prediksi

Tentunya dan machine. Sebelumnya dan supervised. Machine learning dan supervised. Sebelumnya kita punya 100 data. 100 data tersebutnya kayak dan cukup sebagainya. 100 data tersebut kayak kita punya kayak contoh-contoh kita. Lengkap dengan data-nya. Dengan data-nya kita bisa belajar dengan supervised learning. Nah, kita punya juga data yang tidak lengkap. Misalnya kita punya 1 juta data. Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning.

Akurasi vs Interpretasi

Tentunya dan machine. Sebelumnya dan supervised. Machine learning dan supervised. Sebelumnya kita punya 100 data. 100 data tersebutnya kayak dan cukup sebagainya. 100 data tersebut kayak kita punya kayak contoh-contoh kita. Lengkap dengan data-nya. Dengan data-nya kita bisa belajar dengan supervised learning. Nah, kita punya juga data yang tidak lengkap. Misalnya kita punya 1 juta data. Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning.

Machine Learning dengan Python

Untuk yang lagi belajar dasar machine learning

1. Pandas: Untuk preprocessing data
2. Numpy: Untuk preprocessing data
3. Matplotlib: Untuk visualisasi data dan hasil prediksi
4. Scikit-learn: Untuk model machine learning dan hasil prediksi

Untuk yang lagi belajar dasar machine learning dan deep learning

1. Pandas: Untuk preprocessing data
2. Numpy: Untuk preprocessing data
3. Matplotlib: Untuk visualisasi data dan hasil prediksi
4. Scikit-learn: Untuk model machine learning dan hasil prediksi
5. Keras: Untuk deep learning dan hasil prediksi
6. Tensorflow: Untuk deep learning dan hasil prediksi

Tentunya dan machine. Sebelumnya dan supervised. Machine learning dan supervised. Sebelumnya kita punya 100 data. 100 data tersebutnya kayak dan cukup sebagainya. 100 data tersebut kayak kita punya kayak contoh-contoh kita. Lengkap dengan data-nya. Dengan data-nya kita bisa belajar dengan supervised learning. Nah, kita punya juga data yang tidak lengkap. Misalnya kita punya 1 juta data. Nah, kita punya juga unsupervised learning. Nah, kita punya juga semi-supervised learning. Jadi, kita punya 100 data yang lengkap. Kita punya juga 1 juta data yang tidak lengkap. Nah, 100 data yang lengkap tersebut atau 1000 data. Kecuali saya 1000 data yang lengkap di training dengan supervised learning.

Data Pre-Processing

About Preprocessing Datasets

Data Preprocessing adalah proses yang mengubah data mentah ke dalam bentuk yang lebih dapat dipahami. Preprocessing data is a fundamental step in machine learning. Kualitas data dan jumlah informasi useful yang dapat ditemukan dalam kumpulan data sangat penting dalam proses machine learning yang akan kita lakukan.

Preprocessing data is the third step in your Data Science Journey (ini bagian yang ribet, sedikit). Without clean data you'll be having a much harder time seeing the actual important pattern in your exploration.

Data Preprocessing merupakan salah satu yang penting.

According to CrossRiver, data scientists spend 80% of their time organizing and cleaning data!

Why is preprocessing dataset important?

Reasons:

1. It prepares raw, incomplete and unclean data
2. Data Independence is useful
3. Data Reduction is useful for storage, saves a lot on storage
4. Many data scientists can separate the concerns of data wrangling from learning and processing data

Common Problems in Preprocessing Dataset

- Missing Values: Ada beberapa data atau data yang hilang
- Missing Data: Mengisikan data yang hilang atau kosong
- Inconsistent Data: Data yang tidak konsisten
- Data Type Mismatches: Ada data yang salah tipe data (integer, string, koma, dll)
- Noisy Data: Objective ini ini dapatnya Mengurangi data atau
- Data Transformation: Data yang perlu ditransformasikan agar data di olah lebih mudah/cepat
- Dimensional Reduction: Mereduksi dimensi, Misalnya dari 3D ke dalam bentuk 2D/dimensi dua/dimensi dua/dimensi dua
- Feature Selection: Pilih fitur yang mana saja yang akan kita ambil
- Feature Engineering: Bagaimana kita mengolah fitur-fitur yang ada, Misal yang yang bisa digabungkan

Data Pre-Processing

About Preprocessing Datasets

Data Preprocessing adalah proses yang mengubah data mentah ke dalam bentuk yang lebih dapat dipahami.

Preprocessing data is the third step in your Data Science Journey (ini bagian yang ribet, sedikit). Without clean data you'll be having a much harder time seeing the actual important pattern in your exploration.

According to CrossRiver, data scientists spend 80% of their time organizing and cleaning data!

Why is preprocessing dataset important?

Reasons:

1. It prepares raw, incomplete and unclean data
2. Data Independence is useful
3. Data Reduction is useful for storage, saves a lot on storage
4. Many data scientists can separate the concerns of data wrangling from learning and processing data

Common Problems in Preprocessing Dataset

- Missing Values: Ada beberapa data atau data yang hilang
- Missing Data: Mengisikan data yang hilang atau kosong
- Inconsistent Data: Data yang tidak konsisten
- Data Type Mismatches: Ada data yang salah tipe data (integer, string, koma, dll)
- Noisy Data: Objective ini ini dapatnya Mengurangi data atau
- Data Transformation: Data yang perlu ditransformasikan agar data di olah lebih mudah/cepat
- Dimensional Reduction: Mereduksi dimensi, Misalnya dari 3D ke dalam bentuk 2D/dimensi dua/dimensi dua/dimensi dua
- Feature Selection: Pilih fitur yang mana saja yang akan kita ambil
- Feature Engineering: Bagaimana kita mengolah fitur-fitur yang ada, Misal yang yang bisa digabungkan

Duplicate Data

```
df.drop_duplicates(subset=['column1', 'column2', 'column3'], keep='first', inplace=False)
```

Parsing Duplicate Data

```
df.drop_duplicates(subset = 'column1')
```

Menghapus data duplikat hanya pada kolom 1

```
df.drop_duplicates(subset = 'column1', keep = 'last')
```

Menghapus data yang ditemukan dan hanya data terakhir data pada kolom 1 saja yang ditampilkan.

```
df.drop_duplicates(subset = 'column1', keep = False)
```

Menghapus data duplikat hanya pada kolom 1

```
df.drop_duplicates(inplace = True)
```

Menghapus data yang ditemukan dan hasil dari fitur tersebut data pada kolom tidak akan ditampilkan.

Missing Value

Why missing value exist

1. Value yang tidak ada/ tidak ada sesuatu/ada apapun
2. Value ada sebelum ada/ada
3. Value ada/ada
4. Manual/ketikan ada/ada
5. Value ada/ada/ada/ada

Cara Menghandle Missing Value di Pandas

```
df.isnull()
df.isnull().sum()
df.dropna(axis=0, inplace=True)
df.dropna(axis=1, inplace=True)
```

```
df.fillna(df.mean())
df.fillna(df.mean()['column1'])
df.fillna(df.mean()['column1'].astype('int'))
```

```
df.fillna(df.mean()['column1'].astype('int'))
```

```
df.fillna(df.mean()['column1'].astype('int'))
```

```
```

Datetime

The `Datetime` module supplies classes for manipulating dates and times.

```
from datetime import datetime as dt
print(dt.now())
2021-11-30 11:51:36.748272
```

Current string to datetime

```
from datetime import datetime as dt
date = '2021-01-01'
date = dt.strptime(date, '%Y-%m-%')
print(date)
type(date)
2021-01-01 00:00:00
<class 'datetime.datetime'>
```

date_year	date_month	date_day
2019	5	1

Apply

```
DataFrame.apply(arg[, axis, broadcast, raw, reduce, args=(), kwds=kwds])
Apply a function along an axis of the DataFrame.
```

```
df                                    df.apply(lambda x: x**2)
   Income  Income2                    Income  Income2
0   10000    10000                 0  100000 100000000
1   20000    20000                 1  400000 400000000
2   30000    30000                 2  900000 900000000
3   40000    40000                 3 1600000 1600000000
4   50000    50000                 4 2500000 2500000000
```

Standardization (Feature Scaling)

Standardization of datasets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data (for example with zero mean and unit variance).

Making training data (Bivariate) one-class-SVM (novelty detection)

```
# Generate training data
X = 0.3 * np.random.randn(n_samples, 2)
X = np.r_[X + 2, X - 2]
# Generate some regular novel observations
X = 0.3 * np.random.randn(n_samples, 2)
X_test = np.r_[X + 2, X - 2]
# Generate some abnormal novel observations
X_outliers = np.random.uniform(low=-4, high=4,
                               size=(n_samples_test, 2))
```

$$Z = \frac{X - \mu}{\sigma}$$

```
class StandardScaler(BaseEstimator, TransformerMixin):
    with_mean = True
    with_std = True
    copy = True
    def __init__(self, copy=True, with_mean=True, with_std=True):
        self.with_mean = with_mean
        self.with_std = with_std
        self.copy = copy
    def partial_fit(self, X, y=None):
        if hasattr(self, 'scale_'):
            return self.fit(X, y)
        X = check_array(X, accept_sparse=('csr', 'csc'),
                        copy=self.copy,
                        ensure_2d=False, warn_on_dtype=True,
                        estimator=self, dtype=FLOAT_DTYPES)
        if X.dtype == np.float32:
            X = X.astype(np.float, copy=False)
        if self.with_std:
            self.scale_ = _handle_zeros_in_scale(np.sqrt(X.var(axis=0)))
        else:
            self.scale_ = None
        return self
    def fit(self, X, y=None):
        X = check_array(X, accept_sparse=('csr', 'csc'),
                        copy=self.copy,
                        ensure_2d=False, warn_on_dtype=True,
                        estimator=self, dtype=FLOAT_DTYPES)
        if X.dtype == np.float32:
            X = X.astype(np.float, copy=False)
        if self.with_mean:
            mean_ = np.mean(X, axis=0)
        if self.with_std:
            scale_ = _handle_zeros_in_scale(np.sqrt(X.var(axis=0)))
        else:
            scale_ = None
        self.mean_ = mean_
        self.scale_ = scale_
        return self
    def transform(self, X, y=copy):
        check_is_fitted(self, 'scale_')
        X = check_array(X, accept_sparse=('csr', 'csc'),
                        copy=self.copy,
                        ensure_2d=False, warn_on_dtype=True,
                        estimator=self, dtype=FLOAT_DTYPES)
        if X.dtype == np.float32:
            X = X.astype(np.float, copy=False)
        if self.scale_ is not None:
            inplace_column_scale(X, 1/self.scale_)
        else:
            warnings.warn("Mean of all zero data when "
                          "with_std=True, "
                          "cannot be preprocessed "
                          "in a meaningful way.")
        return X
    def inverse_transform(self, X, y=copy):
        check_is_fitted(self, 'scale_')
        X = check_array(X, accept_sparse=('csr', 'csc'),
                        copy=self.copy,
                        ensure_2d=False, warn_on_dtype=True,
                        estimator=self, dtype=FLOAT_DTYPES)
        if X.dtype == np.float32:
            X = X.astype(np.float, copy=False)
        if self.scale_ is not None:
            inplace_column_scale(X, self.scale_)
        else:
            warnings.warn("Mean of all zero data when "
                          "with_std=True, "
                          "cannot be preprocessed "
                          "in a meaningful way.")
        return X
```

Scaling Feature to Range

An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value of each feature is scaled to unit size. This can also be used for centering (sparse).

The of standardized distance having this values. of Minmax scale clean data distortion, this positive value relevant. dif

```
class sklearn.preprocessing.MinMaxScaler(feature_range=(0, 1), copy=True)
```

$$X_{std} = \frac{(X - X_{min})}{(X_{max} - X_{min})}$$

$$X_{scaled} = X_{std} \times (max - min) + min$$

```
>>> from sklearn.preprocessing import MinMaxScaler
>>> data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
>>> scaler = MinMaxScaler()
>>> print(scaler.fit(data))
MinMaxScaler(copy=True, feature_range=(0, 1))
>>> print(scaler.data_max_)
[ 1. 18.]
>>> print(scaler.transform([[2, 12]]))
[[ 1.5 0.5]]
>>> print(scaler.data_min_)
[-1. 2.]
>>> print(scaler.transform([[-2, 1]]))
[[-0.5 -0.05]]
>>> print(scaler.data_range_)
[ 2. 16.]
```

Scaling Range Into 0-1

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
X_train_minmax = scaler.fit_transform(X_train)
print("transformed shape: {}".format(X_train_minmax.shape))
print("per-feature minimum before scaling:\n {}".
      format(X_train.min(axis=0)))
print("per-feature minimum after scaling:\n {}".
      format(X_train_minmax.min(axis=0)))
print("per-feature maximum before scaling:\n {}".
      format(X_train.max(axis=0)))
print("per-feature maximum after scaling:\n {}".
      format(X_train_minmax.max(axis=0)))
```

Scaling Range Into 0-100

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 100))
X_train_minmax = scaler.fit_transform(X_train)
print("transformed shape: {}".format(X_train_minmax.shape))
print("per-feature minimum before scaling:\n {}".
      format(X_train.min(axis=0)))
print("per-feature minimum after scaling:\n {}".
      format(X_train_minmax.min(axis=0)))
print("per-feature maximum before scaling:\n {}".
      format(X_train.max(axis=0)))
print("per-feature maximum after scaling:\n {}".
      format(X_train_minmax.max(axis=0)))
```

Discretization

Those (categorical variable) contains ranksof variable(ordinal). Pandasqcut discretization cuts(binning) equal-sized bins or sample quantiles based on sample quantiles. For example, 5 quantiles will discretize the data into 5 equally populated categories. Discretization is also commonly used to break up continuous data into discrete ranksof (ordinal) for certain (ordinal) (logit) Discretization differentiates known as quantization or binning (provides way to partition continuous features into discrete values. Certain datasets with continuous features may benefit from discretization, because discretization can transform the dataset of continuous attributes to an dataset of only nominal attributes.

```
class sklearn.preprocessing.KBinsDiscretizer(n_bins=5, encode='onehot', strategy='quantile')
```

```
>>> X = [[-2, 1, -4,   1],
>>>      [-1, 2, -3,  -0.5],
>>>      [ 0, 3, -2,   0.5],
>>>      [ 1, 4, -1,   1]]
>>> KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='uniform').
>>> fit_transform(X)
array([[ 0., 0., 0., 1.],
       [ 1., 1., 1., 0.],
       [ 2., 2., 2., 2.],
       [ 2., 2., 2., 2.]])
```

Categorical Encoding (Ordinal Data)

Encoding Data

Name	Skill
0 Jamie	Expert
1 Danny	Medium
2 Steve	Beginner
3 Jason	Expert
4 Joseph	Expert

```
Data['Skill'].replace(['Beginner','Medium','Expert'],
                      [1,2,3],inplace=True)
Data
```

	Name	Skill
0	Jamie	3
1	Danny	2
2	Steve	1
3	Jason	3
4	Joseph	3

Imbalance Class

Imbalanced classes put "accuracy" out of business. This is a surprisingly common problem in machine learning (specifically in classification), and it takes a little extra thought to deal with it properly.

Missing class logic manipulation: gender age test price. Five months was spent 1000 data plus price 100. Kolmo.bus train machine learning. Kolmogorov complexity. Kolmogorov complexity gender age test price. Five months was spent 1000 data plus price 100. Kolmo.bus train machine learning.

Terdapat 2 cara untuk untuk mengatasi imbalance class :

1. Undersampling, cara ini akan mengurangi data yang ada pada kelas yang mayoritas sehingga jumlahnya sama dengan data yang ada di kelas yang minoritas.
2. Oversampling, cara ini akan tetap ada data yang ada pada kelas yang minoritas dan ditambah sebanyak jumlah data yang ada pada kelas mayoritas. Oversampling dapat dilakukan dengan cara melakukan duplikasi pada data yang ada pada kelas minoritas.
3. SMOTE. Cara ini hampir sama dengan cara Oversampling. Perbedaan dari oversampling. Perbedaan dari oversampling adalah data sintetis. Data sintetis akan dibuat dengan menggunakan sebuah algoritma yang akan membuat data tetangga terdekat dari data minoritas.

Cara kedua dan ketiga yang akan kita gunakan untuk mengatasi imbalance class adalah oversampling dan undersampling. Dengan cara kedua ini, kita akan membuat data yang baru dengan cara melakukan duplikasi pada data yang ada. Dengan cara ketiga ini, kita akan membuat data yang baru dengan cara membuat data tetangga terdekat dari data yang ada. Dengan cara ketiga ini, kita akan membuat data yang baru dengan cara membuat data tetangga terdekat dari data yang ada.

Data Partition

Hold-out method

Learn More: Scikitlearn

Preprocessing in Text

The Keras library of the most robust preprocessing textual backlog.

- Text Tokenizing (Without spaces, number, etc. Mengubah teks kata yang lebih pendek)
- Text Lemmatization (Mengubah kata yang ada di dalam bentuk dasar/kata dasar. Contoh: Kata yang berbeda yang hanya memberikan makna serupa)
- Text Stemming (Mengubah kata yang ada di dalam bentuk dasar/kata dasar. Contoh: Kata yang berbeda yang hanya memberikan makna serupa)
- Text Lemmatization. Making words into base form. Mengubah kata kata dasar. Contoh: Kata yang berbeda yang hanya memberikan makna serupa)

Bias-Variance Tradeoff

What is Bias & Variance

- Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data. Bias also underfitting when model has high variance (Bias) (high variance (Bias) (high variance (Bias) (