5월 30일 스터디

메뉴얼 보는 법, 빅분기 작업형 3번 기초 이해하기

https://www.youtube.com/watch?v=_GIBVt5-khk&list=PLSIDi2AkDv82Qv7B3WiWypQSFmOCb-G_-&index=4 (https://www.youtube.com/watch?v=_GIBVt5-khk&list=PLSIDi2AkDv82Qv7B3WiWypQSFmOCb-G_-&index=4)

코드 출처: https://github.com/lovedlim/BigDataCertificationCourses/ (https://github.com/lovedlim/BigDataCertificationCourses/)

https://www.kaggle.com/code/agileteam/t2-exercise-tutorial-baseline/notebook (https://www.kaggle.com/code/agileteam/t2-exercise-tutorial-baseline/notebook)

실제 시험 체험 사이트 : https://dataq.goorm.io/exam/116674/%EC%B2%B4%ED%97%98%ED%95%98%EA%B8%
(https://dataq.goorm.io/exam/116674/%EC%B2%B4%ED%97%98%ED%95%98%EA%B8%

문제를 잘 읽고 문제의 포맷을 그대로 제출해야 함

In [8]:
```
import pandas as pd
```

In [9]:
```
X = pd.read_csv("./X_train.csv", encoding="euc-kr")
y = pd.read_csv("./y_train.csv")

test = pd.read_csv("./X_test.csv", encoding="euc-kr")
```

1) 데이터 탐색

In [10]:
```
X.head()
```

Out[10]:

	cust_id	총구매액	최대구매액	환불금액	주구매상품	주구매지점	내점일수	내점당구매건수	주말방문비율	구매주기
0	0	68282840	11264000	6860000.0	기타	강남점	19	3.894737	0.527027	17
1	1	2136000	2136000	300000.0	스포츠	잠실점	2	1.500000	0.000000	1
2	2	3197000	1639000	NaN	남성캐주얼	관악점	2	2.000000	0.000000	1
3	3	16077620	4935000	NaN	기타	광주점	18	2.444444	0.318182	16
4	4	29050000	24000000	NaN	보석	본 점	2	1.500000	0.000000	85

In [11]:
```
y.head()
```

Out[11]:

	cust_id	gender
0	0	0
1	1	0
2	2	1
3	3	1
4	4	0

In [12]:
```
test.head()
```

Out[12]:

	cust_id	총구매액	최대구매액	환불금액	주구매상품	주구매지점	내점일수	내점당구매건수	주말방문비율	구매주기
0	3500	70900400	22000000	4050000.0	골프	부산본점	13	1.461538	0.789474	26
1	3501	310533100	38558000	48034700.0	농산물	잠실점	90	2.433333	0.369863	3
2	3502	305264140	14825000	30521000.0	가공식품	본점	101	14.623762	0.083277	3
3	3503	7594080	5225000	NaN	주방용품	부산본점	5	2.000000	0.000000	47
4	3504	1795790	1411200	NaN	수산품	청량리점	3	2.666667	0.125000	8

In [13]:
```
X_train.shape, y_train.shape, X_test.shape
```

Out[13]:
```
((3500, 10), (3500, 2), (2482, 10))
```

In [14]:
```
# 결측치 확인
X.isnull().sum()
```

Out[14]:
```
cust_id        0
총구매액           0
최대구매액          0
환불금액        2295
주구매상품          0
주구매지점          0
내점일수           0
내점당구매건수        0
주말방문비율         0
구매주기           0
dtype: int64
```

In [16]:
```
y.isnull().sum()
```

Out[16]:
```
cust_id    0
gender     0
dtype: int64
```

In [17]:
```
test.isnull().sum()
```

Out[17]:
```
cust_id        0
총구매액           0
최대구매액          0
환불금액        1611
주구매상품          0
주구매지점          0
내점일수           0
내점당구매건수        0
주말방문비율         0
구매주기           0
dtype: int64
```

In [18]:
```
X.describe()
```

Out[18]:

	cust_id	총구매액	최대구매액	환불금액	내점일수	내점당구매건수	
count	3500.000000	3.500000e+03	3.500000e+03	1.205000e+03	3500.000000	3500.000000	35
mean	1749.500000	9.191925e+07	1.966424e+07	2.407822e+07	19.253714	2.834963	
std	1010.507298	1.635065e+08	3.199235e+07	4.746453e+07	27.174942	1.912368	
min	0.000000	-5.242152e+07	-2.992000e+06	5.600000e+03	1.000000	1.000000	
25%	874.750000	4.747050e+06	2.875000e+06	2.259000e+06	2.000000	1.666667	
50%	1749.500000	2.822270e+07	9.837000e+06	7.392000e+06	8.000000	2.333333	
75%	2624.250000	1.065079e+08	2.296250e+07	2.412000e+07	25.000000	3.375000	
max	3499.000000	2.323180e+09	7.066290e+08	5.637530e+08	285.000000	22.083333	

In [19]:
```
X.describe(include = "object")
```

Out[19]:

	주구매상품	주구매지점
count	3500	3500
unique	42	24
top	기타	본 점
freq	595	1077

In [20]:
```
test.describe()
```

Out[20]:

	cust_id	총구매액	최대구매액	환불금액	내점일수	내점당구매건수	
count	2482.000000	2.482000e+03	2.482000e+03	8.710000e+02	2482.000000	2482.000000	24
mean	4740.500000	1.010275e+08	2.177048e+07	2.554716e+07	19.516922	2.819388	
std	716.636007	1.732132e+08	3.504919e+07	5.944074e+07	25.973972	1.754550	
min	3500.000000	-3.744000e+07	-3.744000e+07	1.000000e+04	1.000000	1.000000	
25%	4120.250000	5.076868e+06	2.884350e+06	2.414000e+06	2.000000	1.750000	
50%	4740.500000	3.051686e+07	1.075250e+07	8.100000e+06	9.000000	2.430952	
75%	5360.750000	1.264255e+08	2.627700e+07	2.228090e+07	26.750000	3.375000	
max	5981.000000	2.861238e+09	5.932250e+08	8.715144e+08	222.000000	15.875000	

In [21]:
```
test.describe(include = "object")
```

Out[21]:

	주구매상품	주구매지점
count	2482	2482
unique	41	24
top	기타	본 점
freq	465	726

In [23]:
```
y.describe()
```

Out[23]:

	cust_id	gender
count	3500.000000	3500.000000
mean	1749.500000	0.376000
std	1010.507298	0.484449
min	0.000000	0.000000
25%	874.750000	0.000000
50%	1749.500000	0.000000
75%	2624.250000	1.000000
max	3499.000000	1.000000

In [25]:
```
y['gender'].value_counts()
```

Out[25]:
```
0    2184
1    1316
Name: gender, dtype: int64
```

2) 전처리

In [26]:
```
# 결측치처리
X = X.fillna(0) # 환불금액 0값으로 채움
test = test.fillna(0)
```

In [27]:
```
X.head()
```

Out[27]:

	cust_id	총구매액	최대구매액	환불금액	주구매상품	주구매지점	내점일수	내점당구매건수	주말방문비율	구매주기
0	0	68282840	11264000	6860000.0	기타	강남점	19	3.894737	0.527027	17
1	1	2136000	2136000	300000.0	스포츠	잠실점	2	1.500000	0.000000	1
2	2	3197000	1639000	0.0	남성캐주얼	관악점	2	2.000000	0.000000	1
3	3	16077620	4935000	0.0	기타	광주점	18	2.444444	0.318182	16
4	4	29050000	24000000	0.0	보석	본 점	2	1.500000	0.000000	85

In [28]:
```
X = X.drop(['cust_id'], axis=1)
cust_id = test.pop('cust_id')
```

3) 피처 엔지니어링

In [29]:
```
# Label Encoding (범주형 변수 레이블인코딩)
from sklearn.preprocessing import LabelEncoder
cols = ['주구매상품', '주구매지점']
for col in cols:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    test[col] = le.transform(test[col])

X.head()
```

Out[29]:

	총구매액	최대구매액	환불금액	주구매상품	주구매지점	내점일수	내점당구매건수	주말방문비율	구매주기
0	68282840	11264000	6860000.0	5	0	19	3.894737	0.527027	17
1	2136000	2136000	300000.0	21	19	2	1.500000	0.000000	1
2	3197000	1639000	0.0	6	1	2	2.000000	0.000000	1
3	16077620	4935000	0.0	5	2	18	2.444444	0.318182	16
4	29050000	24000000	0.0	15	8	2	1.500000	0.000000	85

4) 모델링 & 하이퍼파라미터 튜닝

In [30]:
```
# 모델링 & 하이퍼파라미터 튜닝 & 앙상블
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=2022)
model.fit(X, y['gender'])
print(model.score(X, y['gender']))
predictions = model.predict_proba(test)
```

```
0.6874285714285714
```

In [31]:
```
predictions[:,1]
```

Out[31]:
```
array([0.43567157, 0.19725558, 0.17732635, ..., 0.43703219, 0.36002886,
       0.54383742])
```

5) 제출

In [32]:
```
# csv생성
output = pd.DataFrame({'cust_id': cust_id, 'gender': predictions[:,1]})
output.head()
```

Out[32]:

	cust_id	gender
0	3500	0.435672
1	3501	0.197256
2	3502	0.177326
3	3503	0.420662
4	3504	0.484252

In [33]:
```
output.to_csv("123456789.csv", index=False)
```

In [34]:
```
pd.read_csv("./123456789.csv").head()
```

Out[34]:

	cust_id	gender
0	3500	0.435672
1	3501	0.197256
2	3502	0.177326
3	3503	0.420662
4	3504	0.484252

In [35]:
```
pd.read_csv("./123456789.csv").tail()
```

Out[35]:

	cust_id	gender
2477	5977	0.474432
2478	5978	0.511048
2479	5979	0.437032
2480	5980	0.360029
2481	5981	0.543837

시험장에서 메뉴얼 보는 방법

https://www.youtube.com/watch?v=2Nf6yAgnZTY&list=PLSIDi2AkDv82Qv7B3WiWypQSFmOCb-G_-&index=13 (https://www.youtube.com/watch?v=2Nf6yAgnZTY&list=PLSIDi2AkDv82Qv7B3WiWypQSFmOCb-G_-&index=13)

https://www.kaggle.com/code/agileteam/tip-guide/notebook (https://www.kaggle.com/code/agileteam/tip-guide/notebook)

반드시 시험장 환경에서 테스트하고 메모장 연습 필요

In [37]:
```
# 함수의 Full Name을 찾는 방법
import pandas as pd
print(dir(pd))
```

```
['ArrowDtype', 'BooleanDtype', 'Categorical', 'CategoricalDtype', 'CategoricalIndex', 'DataFrame', 'DateOffset', 'DatetimeIndex', 'DatetimeTZDtype', 'ExcelFile', 'ExcelWriter', 'Flags', 'Float32Dtype', 'Float64Dtype', 'Float64Index', 'Grouper', 'HDFStore', 'Index', 'IndexSlice', 'Int16Dtype', 'Int32Dtype', 'Int64Dtype', 'Int64Index', 'Int8Dtype', 'Interval', 'IntervalDtype', 'IntervalIndex', 'MultiIndex', 'NA', 'NaT', 'NamedAgg', 'Period', 'PeriodDtype', 'PeriodIndex', 'RangeIndex', 'Series', 'SparseDtype', 'StringDtype', 'Timedelta', 'TimedeltaIndex', 'Timestamp', 'UInt16Dtype', 'UInt32Dtype', 'UInt64Dtype', 'UInt64Index', 'UInt8Dtype', '__all__', '__builtins__', '__cached__', '__deprecated_num_index_names', '__dir__', '__doc__', '__docformat__', '__file__', '__getattr__', '__git_version__', '__loader__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_config', '_is_numpy_dev', '_libs', '_testing', '_typing', '_version', 'annotations', 'api', 'array', 'arrays', 'bdate_range', 'compat', 'concat', 'core', 'crosstab', 'cut', 'date_range', 'describe_option', 'errors', 'eval', 'factorize', 'from_dummies', 'get_dummies', 'get_option', 'infer_freq', 'interval_range', 'io', 'isna', 'isnull', 'json_normalize', 'lreshape', 'melt', 'merge', 'merge_asof', 'merge_ordered', 'notna', 'notnull', 'offsets', 'option_context', 'options', 'pandas', 'period_range', 'pivot', 'pivot_table', 'plotting', 'qcut', 'read_clipboard', 'read_csv', 'read_excel', 'read_feather', 'read_fwf', 'read_gbq', 'read_hdf', 'read_html', 'read_json', 'read_orc', 'read_parquet', 'read_pickle', 'read_sas', 'read_spss', 'read_sql', 'read_sql_query', 'read_sql_table', 'read_stata', 'read_table', 'read_xml', 'reset_option', 'set_eng_float_format', 'set_option', 'show_versions', 'test', 'testing', 'timedelta_range', 'to_datetime', 'to_numeric', 'to_pickle', 'to_timedelta', 'tseries', 'unique', 'util', 'value_counts', 'wide_to_long']
```

In [38]:
```
#데이터 프레임에서 할 수 있는 것들은?
print(dir(pd.DataFrame))
```

['T', '_AXIS_LEN', '_AXIS_NAMES', '_AXIS_NUMBERS', '_AXIS_ORDERS', '_AXIS_TO_AXIS_
NUMBER', '_HANDLED_TYPES', '__abs__', '__add__', '__and__', '__annotations__', '__
array__', '__array_priority__', '__array_ufunc__', '__array_wrap__', '__bool__',
'__class__', '__contains__', '__copy__', '__dataframe__', '__deepcopy__', '__delat
tr__', '__delitem__', '__dict__', '__dir__', '__divmod__', '__doc__', '__eq__', '_
_finalize__', '__floordiv__', '__format__', '__ge__', '__getattr__', '__getattribu
te__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__iand_
_', '__ifloordiv__', '__imod__', '__imul__', '__init__', '__init_subclass__', '__i
nvert__', '__ior__', '__ipow__', '__isub__', '__iter__', '__itruediv__', '__ixor_
_', '__le__', '__len__', '__lt__', '__matmul__', '__mod__', '__module__', '__mul_
_', '__ne__', '__neg__', '__new__', '__nonzero__', '__or__', '__pos__', '__pow__',
'__radd__', '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__',
'__rfloordiv__', '__rmatmul__', '__rmod__', '__rmul__', '__ror__', '__round__', '_
_rpow__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__setitem__', '_
_setstate__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv_
_', '__weakref__', '__xor__', '_accessors', '_accum_func', '_add_numeric_operation
s', '_agg_by_level', '_agg_examples_doc', '_agg_summary_and_see_also_doc', '_align
_frame', '_align_series', '_append', '_arith_method', '_as_manager', '_box_col_val
ues', '_can_fast_transpose', '_check_inplace_and_allows_duplicate_labels', '_check
_inplace_setting', '_check_is_chained_assignment_possible', '_check_label_or_level
_ambiguity', '_check_setitem_copy', '_clear_item_cache', '_clip_with_one_bound',
'_clip_with_scalar', '_cmp_method', '_combine_frame', '_consolidate', '_consolidat
e_inplace', '_construct_axes_dict', '_construct_axes_from_arguments', '_construct_
result', '_constructor', '_constructor_sliced', '_convert', '_count_level', '_dat
a', '_dir_additions', '_dir_deletions', '_dispatch_frame_op', '_drop_axis', '_drop
_labels_or_levels', '_ensure_valid_index', '_find_valid_index', '_from_arrays', '_
get_agg_axis', '_get_axis', '_get_axis_name', '_get_axis_number', '_get_axis_resol
vers', '_get_block_manager_axis', '_get_bool_data', '_get_cleaned_column_resolver
s', '_get_column_array', '_get_index_resolvers', '_get_item_cache', '_get_label_or
_level_values', '_get_numeric_data', '_get_value', '_getitem_bool_array', '_getite
m_multilevel', '_gotitem', '_hidden_attrs', '_indexed_same', '_info_axis', '_info_
axis_name', '_info_axis_number', '_info_repr', '_init_mgr', '_inplace_method', '_i
nternal_names', '_internal_names_set', '_is_copy', '_is_homogeneous_type', '_is_la
bel_or_level_reference', '_is_label_reference', '_is_level_reference', '_is_mixed_
type', '_is_view', '_iset_item', '_iset_item_mgr', '_iset_not_inplace', '_iter_col
umn_arrays', '_ixs', '_join_compat', '_logical_func', '_logical_method', '_maybe_c
ache_changed', '_maybe_update_cacher', '_metadata', '_min_count_stat_function', '_
needs_reindex_multi', '_protect_consolidate', '_reduce', '_reduce_axis1', '_reinde
x_axes', '_reindex_columns', '_reindex_index', '_reindex_multi', '_reindex_with_in
dexers', '_rename', '_replace_columnwise', '_repr_data_resource_', '_repr_fits_hor
izontal_', '_repr_fits_vertical_', '_repr_html_', '_repr_latex_', '_reset_cache',
'_reset_cacher', '_sanitize_column', '_series', '_set_axis', '_set_axis_name', '_s
et_axis_nocheck', '_set_is_copy', '_set_item', '_set_item_frame_value', '_set_item
_mgr', '_set_value', '_setitem_array', '_setitem_frame', '_setitem_slice', '_slic
e', '_stat_axis', '_stat_axis_name', '_stat_axis_number', '_stat_function', '_stat
_function_ddof', '_take', '_take_with_is_copy', '_to_dict_of_blocks', '_typ', '_up
date_inplace', '_validate_dtype', '_values', '_where', 'abs', 'add', 'add_prefix',
'add_suffix', 'agg', 'aggregate', 'align', 'all', 'any', 'append', 'apply', 'apply
map', 'asfreq', 'asof', 'assign', 'astype', 'at', 'at_time', 'attrs', 'axes', 'bac
kfill', 'between_time', 'bfill', 'bool', 'boxplot', 'clip', 'columns', 'combine',
'combine_first', 'compare', 'convert_dtypes', 'copy', 'corr', 'corrwith', 'count',
'cov', 'cummax', 'cummin', 'cumprod', 'cumsum', 'describe', 'diff', 'div', 'divid
e', 'dot', 'drop', 'drop_duplicates', 'droplevel', 'dropna', 'dtypes', 'duplicate
d', 'empty', 'eq', 'equals', 'eval', 'ewm', 'expanding', 'explode', 'ffill', 'fill
na', 'filter', 'first', 'first_valid_index', 'flags', 'floordiv', 'from_dict', 'fr
om_records', 'ge', 'get', 'groupby', 'gt', 'head', 'hist', 'iat', 'idxmax', 'idxmi
n', 'iloc', 'index', 'infer_objects', 'info', 'insert', 'interpolate', 'isetitem',

```
'isin', 'isna', 'isnull', 'items', 'iteritems', 'iterrows', 'itertuples', 'join',
'keys', 'kurt', 'kurtosis', 'last', 'last_valid_index', 'le', 'loc', 'lookup', 'l
t', 'mad', 'mask', 'max', 'mean', 'median', 'melt', 'memory_usage', 'merge', 'mi
n', 'mod', 'mode', 'mul', 'multiply', 'ndim', 'ne', 'nlargest', 'notna', 'notnul
l', 'nsmallest', 'nunique', 'pad', 'pct_change', 'pipe', 'pivot', 'pivot_table',
'plot', 'pop', 'pow', 'prod', 'product', 'quantile', 'query', 'radd', 'rank', 'rdi
v', 'reindex', 'reindex_like', 'rename', 'rename_axis', 'reorder_levels', 'replac
e', 'resample', 'reset_index', 'rfloordiv', 'rmod', 'rmul', 'rolling', 'round', 'r
pow', 'rsub', 'rtruediv', 'sample', 'select_dtypes', 'sem', 'set_axis', 'set_flag
s', 'set_index', 'shape', 'shift', 'size', 'skew', 'slice_shift', 'sort_index', 's
ort_values', 'sparse', 'squeeze', 'stack', 'std', 'style', 'sub', 'subtract', 'su
m', 'swapaxes', 'swaplevel', 'tail', 'take', 'to_clipboard', 'to_csv', 'to_dict',
'to_excel', 'to_feather', 'to_gbq', 'to_hdf', 'to_html', 'to_json', 'to_latex', 't
o_markdown', 'to_numpy', 'to_orc', 'to_parquet', 'to_period', 'to_pickle', 'to_rec
ords', 'to_sql', 'to_stata', 'to_string', 'to_timestamp', 'to_xarray', 'to_xml',
'transform', 'transpose', 'truediv', 'truncate', 'tshift', 'tz_convert', 'tz_local
ize', 'unstack', 'update', 'value_counts', 'values', 'var', 'where', 'xs']
```

In [39]:
```
#help을 통해 사용법 확인
# 데이터 프레임에서 결측치 drop을 어떻게 사용했더라?
print(help(pd.DataFrame.drop))
```

```
Help on function drop in module pandas.core.frame:

drop(self, labels: 'IndexLabel' = None, *, axis: 'Axis' = 0, index: 'IndexLabel'
= None, columns: 'IndexLabel' = None, level: 'Level' = None, inplace: 'bool' = F
alse, errors: 'IgnoreRaise' = 'raise') -> 'DataFrame | None'
    Drop specified labels from rows or columns.

    Remove rows or columns by specifying label names and corresponding
    axis, or by specifying directly index or column names. When using a
    multi-index, labels on different levels can be removed by specifying
    the level. See the `user guide <advanced.shown_levels>`
    for more information about the now unused levels.

    Parameters
    ----------
    labels : single label or list-like
        Index or column labels to drop. A tuple will be used as a single
        label and not treated as a list-like.
    axis : {0 or 'index', 1 or 'columns'}, default 0
        Whether to drop labels from the index (0 or 'index') or
```

In [40]:
```
# 예를 들어, 원핫인코딩 어떻게 사용했더라?
import pandas as pd
print(help(pd.get_dummies))
```

```
Help on function get_dummies in module pandas.core.reshape.encoding:

get_dummies(data, prefix=None, prefix_sep='_', dummy_na: 'bool' = False, columns=N
one, sparse: 'bool' = False, drop_first: 'bool' = False, dtype: 'Dtype | None' = N
one) -> 'DataFrame'
    Convert categorical variable into dummy/indicator variables.

    Parameters
    ----------
    data : array-like, Series, or DataFrame
        Data of which to get dummy indicators.
    prefix : str, list of str, or dict of str, default None
        String to append DataFrame column names.
        Pass a list with length equal to the number of columns
        when calling get_dummies on a DataFrame. Alternatively, `prefix`
        can be a dictionary mapping column names to prefixes.
    prefix_sep : str, default '_'
        If appending prefix, separator/delimiter to use. Or pass a
        list or dictionary as with `prefix`.
    dummy_na : bool, default False
        Add a column to indicate NaNs, if False NaNs are ignored.
    columns : list-like, default None
        Column names in the DataFrame to be encoded.
        If `columns` is None then all the columns with
        `object`, `string`, or `category` dtype will be converted.
    sparse : bool, default False
        Whether the dummy-encoded columns should be backed by
        a :class:`SparseArray` (True) or a regular NumPy array (False).
    drop_first : bool, default False
        Whether to get k-1 dummies out of k categorical levels by removing the
        first level.
    dtype : dtype, default np.uint8
        Data type for new columns. Only a single dtype is allowed.

    Returns
    -------
    DataFrame
        Dummy-coded data.

    See Also
    --------
    Series.str.get_dummies : Convert Series to dummy codes.
    :func:`~pandas.from_dummies` : Convert dummy codes to categorical ``DataFrame`
`.

    Notes
    -----
    Reference :ref:`the user guide <reshaping.dummies>` for more examples.

    Examples
    --------
    >>> s = pd.Series(list('abca'))

    >>> pd.get_dummies(s)
       a  b  c
    0  1  0  0
    1  0  1  0
```

```
2  0  0  1
3  1  0  0

>>> s1 = ['a', 'b', np.nan]

>>> pd.get_dummies(s1)
   a  b
0  1  0
1  0  1
2  0  0

>>> pd.get_dummies(s1, dummy_na=True)
   a  b  NaN
0  1  0    0
1  0  1    0
2  0  0    1

>>> df = pd.DataFrame({'A': ['a', 'b', 'a'], 'B': ['b', 'a', 'c'],
...                    'C': [1, 2, 3]})

>>> pd.get_dummies(df, prefix=['col1', 'col2'])
   C  col1_a  col1_b  col2_a  col2_b  col2_c
0  1       1       0       0       1       0
1  2       0       1       1       0       0
2  3       1       0       0       0       1

>>> pd.get_dummies(pd.Series(list('abcaa')))
   a  b  c
0  1  0  0
1  0  1  0
2  0  0  1
3  1  0  0
4  1  0  0

>>> pd.get_dummies(pd.Series(list('abcaa')), drop_first=True)
   b  c
0  0  0
1  1  0
2  0  1
3  0  0
4  0  0

>>> pd.get_dummies(pd.Series(list('abc')), dtype=float)
     a    b    c
0  1.0  0.0  0.0
1  0.0  1.0  0.0
2  0.0  0.0  1.0
```

None

In [41]:
```
# skleanr 은 약간 다름
import sklearn
print(sklearn.__all__)
```

```
['calibration', 'cluster', 'covariance', 'cross_decomposition', 'datasets', 'decom
position', 'dummy', 'ensemble', 'exceptions', 'experimental', 'externals', 'featur
e_extraction', 'feature_selection', 'gaussian_process', 'inspection', 'isotonic',
'kernel_approximation', 'kernel_ridge', 'linear_model', 'manifold', 'metrics', 'mi
xture', 'model_selection', 'multiclass', 'multioutput', 'naive_bayes', 'neighbor
s', 'neural_network', 'pipeline', 'preprocessing', 'random_projection', 'semi_supe
rvised', 'svm', 'tree', 'discriminant_analysis', 'impute', 'compose', 'clone', 'ge
t_config', 'set_config', 'config_context', 'show_versions']
```

In [42]:
```
# 전처리 무엇을 할 수 있지?
import sklearn.preprocessing
print(sklearn.preprocessing.__all__)
```

```
['Binarizer', 'FunctionTransformer', 'KBinsDiscretizer', 'KernelCenterer', 'LabelB
inarizer', 'LabelEncoder', 'MultiLabelBinarizer', 'MinMaxScaler', 'MaxAbsScaler',
'QuantileTransformer', 'Normalizer', 'OneHotEncoder', 'OrdinalEncoder', 'PowerTran
sformer', 'RobustScaler', 'SplineTransformer', 'StandardScaler', 'add_dummy_featur
e', 'PolynomialFeatures', 'binarize', 'normalize', 'scale', 'robust_scale', 'maxab
s_scale', 'minmax_scale', 'label_binarize', 'quantile_transform', 'power_transfor
m']
```

In [43]:
```
# 문제에서 민맥스스케일을 적용하라고 하네. 어떻게 사용하지?
import sklearn.preprocessing
print(help(sklearn.preprocessing.MinMaxScaler))
```

```
Help on class MinMaxScaler in module sklearn.preprocessing._data:

class MinMaxScaler(sklearn.base._OneToOneFeatureMixin, sklearn.base.TransformerM
ixin, sklearn.base.BaseEstimator)
 |  MinMaxScaler(feature_range=(0, 1), *, copy=True, clip=False)
 |
 |  Transform features by scaling each feature to a given range.
 |
 |  This estimator scales and translates each feature individually such
 |  that it is in the given range on the training set, e.g. between
 |  zero and one.
 |
 |  The transformation is given by::
 |
 |      X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
 |      X_scaled = X_std * (max - min) + min
 |
 |  where min, max = feature_range.
 |
 |  This transformation is often used as an alternative to zero mean
```

In [44]:
```
# 데이터를 나눠야 하는데 풀네임이 뭐더라?
# 데이터를 트레인과 테스트로 나눠야할 때 model_selection안에 있다는건 아셔야 해요^^
import sklearn.model_selection
print(sklearn.model_selection.__all__)
```

```
['BaseCrossValidator', 'BaseShuffleSplit', 'GridSearchCV', 'TimeSeriesSplit', 'KFo
ld', 'GroupKFold', 'GroupShuffleSplit', 'LeaveOneGroupOut', 'LeaveOneOut', 'LeaveP
GroupsOut', 'LeavePOut', 'RepeatedKFold', 'RepeatedStratifiedKFold', 'ParameterGri
d', 'ParameterSampler', 'PredefinedSplit', 'RandomizedSearchCV', 'ShuffleSplit',
'StratifiedKFold', 'StratifiedGroupKFold', 'StratifiedShuffleSplit', 'check_cv',
'cross_val_predict', 'cross_val_score', 'cross_validate', 'learning_curve', 'permu
tation_test_score', 'train_test_split', 'validation_curve']
```

In [45]:
```
# 어떻게 사용하더라??
import sklearn.model_selection
print(help(sklearn.model_selection.train_test_split))
```

```
Help on function train_test_split in module sklearn.model_selection._split:

train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuf
fle=True, stratify=None)
    Split arrays or matrices into random train and test subsets.

    Quick utility that wraps input validation and
    ``next(ShuffleSplit().split(X, y))`` and application to input data
    into a single call for splitting (and optionally subsampling) data in a
    oneliner.

    Read more in the :ref:`User Guide <cross_validation>`.

    Parameters
    ----------
    *arrays : sequence of indexables with same length / shape[0]
        Allowed inputs are lists, numpy arrays, scipy-sparse
        matrices or pandas dataframes.

    test_size : float or int, default=None
        If float, should be between 0.0 and 1.0 and represent the proportion
        of the dataset to include in the test split. If int, represents the
        absolute number of test samples. If None, the value is set to the
        complement of the train size. If ``train_size`` is also None, it will
        be set to 0.25.

    train_size : float or int, default=None
        If float, should be between 0.0 and 1.0 and represent the
        proportion of the dataset to include in the train split. If
        int, represents the absolute number of train samples. If None,
        the value is automatically set to the complement of the test size.

    random_state : int, RandomState instance or None, default=None
        Controls the shuffling applied to the data before applying the split.
        Pass an int for reproducible output across multiple function calls.
        See :term:`Glossary <random_state>`.

    shuffle : bool, default=True
        Whether or not to shuffle the data before splitting. If shuffle=False
        then stratify must be None.

    stratify : array-like, default=None
        If not None, data is split in a stratified fashion, using this as
        the class labels.
        Read more in the :ref:`User Guide <stratification>`.

    Returns
    -------
    splitting : list, length=2 * len(arrays)
        List containing train-test split of inputs.

        .. versionadded:: 0.16
            If the input is sparse, the output will be a
            ``scipy.sparse.csr_matrix``. Else, output type is the same as the
            input type.

    Examples
```

```
--------
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]

>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]

>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]
```

None

In [46]:
```
# 앙상블 모델 쓸래!
import sklearn.ensemble
print(sklearn.ensemble.__all__)
```

```
['BaseEnsemble', 'RandomForestClassifier', 'RandomForestRegressor', 'RandomTreesEm
bedding', 'ExtraTreesClassifier', 'ExtraTreesRegressor', 'BaggingClassifier', 'Bag
gingRegressor', 'IsolationForest', 'GradientBoostingClassifier', 'GradientBoosting
Regressor', 'AdaBoostClassifier', 'AdaBoostRegressor', 'VotingClassifier', 'Voting
Regressor', 'StackingClassifier', 'StackingRegressor', 'HistGradientBoostingClassi
fier', 'HistGradientBoostingRegressor']
```

In [47]:
```
# 랜덤포레스트 어떻게 썼더라?
import sklearn.ensemble
print(help(sklearn.ensemble.RandomForestClassifier()))
```

```
Help on RandomForestClassifier in module sklearn.ensemble._forest object:

class RandomForestClassifier(ForestClassifier)
 |  RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=Non
e, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_fe
atures='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, o
ob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, cla
ss_weight=None, ccp_alpha=0.0, max_samples=None)
 |
 |  A random forest classifier.
 |
 |  A random forest is a meta estimator that fits a number of decision tree
 |  classifiers on various sub-samples of the dataset and uses averaging to
 |  improve the predictive accuracy and control over-fitting.
 |  The sub-sample size is controlled with the `max_samples` parameter if
 |  `bootstrap=True` (default), otherwise the whole dataset is used to build
 |  each tree.
 |
 |  Read more in the :ref:`User Guide <forest>`.
 |
```

In [ ]: