**데이터 전처리, EDA**

**Numpy**

ndarray내에는 같은 데이터 타입만 가능함.

만약 데이터 유형이 섞이면 크기가 더 큰 데이터 타입으로 형이 일괄 변환됨.

float를 int로 변환하면 소수점은 버림.

사이킷런,텐서플로에서 판다스의 df를 입력으로 받을 수 있지만, 기본적으로 넘파이를 입력으로 받음.

넘파이는 슬라이싱, 불린, 팬시만 통함.

넘파이는 자동으로 ‘.iloc’가 붙어있다고 생각하면 편하다.

**변환(Transform)**

ndarray.dtype(‘int32’) # 데이터 타입 변환

ndarray.tolist() # ndarray -> list 변환

ndarray.reshape(row, col, height) # ndarray,list 모양 변경

np.reshape(ndarray,(row, col, height))

a[row] = 0 # 2차원 ndarray에서 행 일괄 변환

a[:, col] = 0 # 2차원 ndarray에서 열 일괄 변환

**생성(Definition)**

np.arange(1,10,2) # 1~9까지 홀수 리스트 생성

np.zeros((2,3), dtype=’int32’) # (2,3) 모양의 0 ndarray 생성

np.dot() # 행렬 내적

np.transpose() # 전치 행렬 생성

np.linspace(start, end, n) # s에서 end까지 균등하게 n개 값 생성

np.random.random() # 0~1사이 실수 생성

np.random.randint(start, end, shape or num) # s에서 end까지 n개 생성

np.random.choice(list, shape or num) # list or ndarray에서 무작위 n개 추출.

random.sample(list, k) # list에서 무작위 k개 추출 (중복x)

np.unique(ndarray) # ndarray 내 인자 반환.

Np.var() # 분산 반환

Np.std() # 표준편차 반환

Np.quantitle() # 4분위수 반환

**정렬(Sorting)**

np.sort(list) # 정렬 return o, 원 함수 변경 x

ndarray.sort() # 정렬 return x, 원 함수 변경 o

ndarray[::-1] # 역정렬

np.sort(array, axis=0 or 1) # 행,열 기준 정렬

np.argsort(array) # 정렬 행렬의 원본 행렬 인덱스 생성

ndarray[np.argsort(array)] # 정렬 행렬 생성

**Pandas**

DataFrame은 여러 개의 Series로 이루어져있음.

DF 및 Series의 Index 객체는 함부로 변경할 수 없음.

Df 와 Series의 row는 인덱스로 접근이 가능한거다.

기본적으로 df의 인덱스값은 명칭 기반 인덱싱이다.

**생성(Create)**

Pd.read\_csv(path, sep, sheet\_name, index\_col)

pd.DataFrame(dict) # key는 col으로 value는 row로 df 생성.

pd.DataFrame(list or ndarray, columns=[]) # list, ndarray로부터 df 생성.

pd.DataFrame(np.zeros(df.shape), columns=df.columns) # 프레임만 남은 빈 df 생성

df.copy() # df 복제

df[‘col’]=0 # 일괄 삽입

df[‘new\_col’] = df[‘col1’] \* 10 + df[‘col2’] # 파생변수 생성, 수정

Df.drop(labels = None, axis=0, index, columns, level, inplace) #열 삭제

(axis = 0 : 행 삭제, 이상치 제거, axis = 1 : 특성 제거)

Pd.Series(list, index) # 시리즈 생성.

**변환(Transformation)**

Series.tolist() # Series -> list

Series.to\_numpy() # Series -> ndarray

df.to\_numpy() # df -> ndarray

df.to\_dict() # df -> dict

df[‘col’] = df[‘col’].replace([1,2],[‘male’,’female’]) # series 내 항목 변경

x1\_dum = pd.get\_dummies(x1) # 원-핫 인코딩

**반환(Return)**

df[‘col’].value\_counts() # 열에 포함된 값 카운트 및 정렬해서 반환,

고유 칼럼 값을 식별자로 사용할 수 있음.

df.index # df index 객체

df.index.values # df index

df.columns # df columns

df.values # df values

Series.index # Series.index 객체

Series.index.values # Series.index.

df.count() # df col별 개수 세기.

df.isna() # df 내 결측치 여부 출력.

df.isna().sum() # df 내 col별 결측치 개수 출력.

Df[df.columns.difference([‘col1’,’col2’]) # df 내 원하는 칼럼만 반환

Df.duplicated([‘col1’, keep=’first’ or ‘last’ or ‘false’] # df 내 중복이 있으면 True

Df.drop\_duplicates([‘col’], keep) # df 내 중복값 처리

**고수준(High Level)**

df.reset\_index(inplace=False, drop=False) # 인덱스 초기화

df.rename(columns={‘old’:’new’, ‘old2’:,’new2’} # 열 이름 선택적으로 바꾸기

df.sort\_values(by, ascending, place) # df 정렬

df.groupby(by=[‘col1’, ‘col2’ ]) # df 그룹화

df.groupby().aggregation() # df 그룹화 후 aggr 함수 적용

df.groupby(by=’col1’)[‘col2’].agg([func1,func2]) # df 그룹화 후 특정 col에 대해 다중 함수 적용

df.groupby(by=’col1’).agg({‘Age’:’max’, ‘Fare’:’mean’}) # 그룹화 후 여러 col에 대해 다중함수 적용

Df.dropna(axis, subset=[‘col1’,’col2’], how = ‘any’ or ‘all’, inplace) # 결측치가 있는 행,열 처리

df[‘col’].fillna(values, method, inplace=False) # df 내 col의 결측치 일괄 처리

(method : ‘bfill’, ‘pad’, ‘ffill’)

df.interpolate(method=’linear’, limit\_direction = ‘forward’) # df 내 결측치 선형 보간

lambda x : x\*\*2 # lambda 식 (x는 입력인자, x\*\*2는 반환식)

map(lambda x : x\*\*2, list) # 다중 인자 lambda 식

df[‘new\_col’] = df[‘col’].apply(lambda x : len(x)) # apply lambda 식

df[‘new\_col’] = df[‘col’].apply(lambda x : a if x > n else b) # apply lambda 식 + if식

(케이스가 많을 경우 함수로 정의해서 사용)

**Str, List, Set, Dict**

**str**

str.split(sep, maxsplit) # 문자열 구분

str.strip(value) # 양쪽 불필요한 인자 제거

‘,’.join(list) # 구분자를 기준으로 문자열 병합

str.find(value, start, end) # 특정 문자열의 인덱스 반환

str.count(value, start, end) # 특정 문자열의 개수 반환

str.replace(old\_value, new\_value, count) # 특정 문자열 교체

str.isalpha() # 문자열이 모두 문자면 True

str.isdigit() # 문자열이 모두 숫자면 True

str.lower() # 소문자 반환

str.upper() # 대문자 반환

**list**

list[start : end : step] # 리스트 슬라이싱

‘a’ in list, ‘b’ not in list # 리스트 내 특정항목의 존재 여부 확인

list.append(value) # 리스트에 값 추가

list.insert(index, value) # 특정 위치에 값 추가

list.extend(value) # 리스트에 여러 값을 추가

list.remove(value) # 특정 값 삭제

list.pop() # 가장 끝에 항목 삭제 및 반환

list.index(value) # 특정 값의 인덱스 반환

list.count(value) # 특정 값의 개수 반환

list.sort() # 리스트 정렬

list.reverse() # 리스트 역배치

list = [i\*\*2 for i in range() if i>=3] # 리스트 컴프리핸션

**set**

set(A) # 집합 생성

A.intersection(B) # 교집합 반환

A.union(B) # 합집합 반환

A.difference(B) # 차집합 반환

**dict**

dict = {‘key1’:’val1’, ‘key2’:’val2’} # 사전 생성

dict[‘key1’] = ‘val1’ # 키 추가 or 변경

del dict[‘key1’] # 키 삭제

dict.keys() # 사전의 키 반환

dict.values() # 사전의 값 반환

dict.items() # 사전의 키,값 항목 반환

dict.clear() # 사전 클리어

dict1.update(dict2) # 사전 업데이트 (항목추가)

**Datetime**

Date\_obj = datetime.date(year,month,day) # date 객체 생성

Time\_obj = datetime.time(hout, minute, second) # time 객체 생성

Datetime\_obj = datetime.datetime(year, month, day, hour, minute, second)

Date.year # date 객체에서 year만 추출

Time\_diff = day2 – day1 # timedelta 객체 생성

Time\_diff.days # timedelta에서 days만 추출

Datetime.date.today() # 오늘 날짜 생성.

Datetime.datetime.now() # 현 날짜 및 시각 새성

Pd.date\_range(start, end, periods, freq=’D’) # 일정기간 동안의 datetime 리스트 생성.

**판다스에서 시계열 자료형은 Timestamp = datetime와 Period라는 두 가지 타입이 있다.**

Pd.to\_datetime(series) # datetime : 시계열 객체로 변환

dt.to\_period(freq= [‘D’ or ‘M’ or ‘A’]) # datetime을 주기로 끊어서 반환

dt.dt.year # datetime에서 연도만 반환

dt.dt.month # datetime에서 월만 반환

dt.dt.day # datetime에서 날짜만 반환

**EDA (matplotlib, seaborn)**

%matplotlib qt

% matplotlib inline

Plt.plot(x, y, fmt=’[color][linestyle][marker]’)

Plt.show()

Plt.clf()

Plt.figure(n)

Plt.subplot(row, col, p)

Plt.xlim(min, max)

Plt.ylim(min, max)

Plt.xlabel()

Plt.ylabel()

Plt.title()

Plt.xticks(x, label, rotation)

Plt.yticks(y, label, rotation)

Plt.grid(True)

Plt.legend([], loc)

Plt.text(x,y,str)

Plt.axhline( y, linestyle, color)

Plt.axvline( x, linestyle, color)

Plt.scatter(x,y,s,c,marker,alpha)

Plt.bar(x,height,width,color,label)

Plt.barh(x,height,width,color,label)

Plt.his(list, bins)

Plt.pie(x, labels, autopct=’%.1f%%’, startangle, counterclock, shadow, explode)

Plt.savefig(path,dpi=100)

Matplotlib.rcPrams[‘font.family’] = ‘Malgun Gothic’

Maplotlib.rcParams[‘axes.unicode\_minus’] = False

**Etc**

for a,b in zip(list1, list2): # 두 개의 리스트로부터 인자를 받는 for문

print(‘반지름 : {0:1d}, 원주율 : {1:.4f}”.format(r,pi) # 형식 출력 1

print('반지름 : %1d, 원주율 : %.4f' % (r,pi)) # 형식 출력 2