**What is big data**

Process and techniques to discover from data answers or insights to solve a business problem.

비즈니스 문제를 해결하기 위한 데이터나 견해를 발견하는 기술이나 프로세스

**Big Data Challenges**

Diverse Data, Sources , Types, Semantics

**Application Areas**

Business, governance, security, medicine, Categorization and Summarization of Text

**Requirements for Big Data**

Business needs , Data, Computers and software, team of trained people

**1.Objective Setting**

Example Business Needs, The Limited Role of Big Data

**2.Data Curation**

**Tasks**

Determine the data needed to best meet the business objective.

비즈니스 목적을 가장 잘 충족시키는데 필요한 데이터를 알아내는 것

Collect the data.

데이터를 고르는 것

Store the data in the computer.

컴퓨터에 저장

**Difficulties**

It is difficult to understand the relationship between data and business needs.

데이터와 비즈니스 목적 사이의 관계를 이해하는 것은 어렵다

The data quality has to be acceptable

데이터 품질이 어셉터블해야한다.

The amount of data has to be enough to be statistically significant

데이터의 양이 통계학적으로 유의할 만큼 충분해야한다.

**3.Data Inspection**

**Data exploration** - Explores the data collected to understand the general characteristics of the data 데이터의 일반적인 특징을 이해하기위해 데이터를 골라 explore하는 것

Check description, distribution, outliers, correlation among attributes

**Suitability check** - Confirms suitability of the data for the business objective

Metadata – technical, business, process

**4.Data preprocessing**

**4.1Data Restructuring**

**Table Decomposition**

Horizontal - row

vertical - column

**Table Merge**

**4.2Data Value Changes**

**4.2.1Cleaning dirty data**

**4.2.1.1Missing Data**

**Ignore the feature(column) (vertical decomposition)**

pro: simple, typically not biased

con: may be a very useful feature

**Ignore the sample (row) (horizontal decomposition)**

pro: simple, all features are kept

con: removed samples may be biased

con: data may become small

df.isna().any() – column에 한 개라도 na가 있으면 true

df.isna().sum() – column 별 na 개수의 합

pd.read\_csv(“dataset”,na\_values = missingdatalist)

df.replace({“~를” : np.nan}, inplace = True)

df.dropna(axis=0, how=’all’, inplace) – 디폴트는 any

how = any – 한 개라도 na가 있으면 드랍

how = all – 모두 na 면 드랍

thresh = 숫자 na가 숫자이상 있으면 drop

df.fillna(숫자) 숫자로 na를 채움

df.fillna(axis=0, method = ‘메소드’, limit=숫자)

ffill 이면 앞에서 bfill이면 뒤에서 숫자만큼 간다.

**4.2.1.2Wrong Data**

**DBMS 가 integrity constraint를 지원하지 않아서.**

**Age,height의 경우 170,25 처럼 되는경우**

**4.2.1.3Unusable Data**

**Due to ambiguous meanings 모호한 의미**

Ex) 전라도 경기도 광주

**Due to multiple valid representations 여러 유효한 표시**

Ex) Microsoft와 ms 처럼 같은 의미인데 다르게 사용

**Due to non-conformance to standards 표준 준수하지 않음**

Ex) 전화번호에서 어떤건 -가 붙고 어떤건 안붙고

**Due to database integration 데이터베이스 integration**

Ex) 데이터의 표현이 다르게 저장됨

**Redundant data**

**Two Steps in Dealing with Dirty Data**

**(1)Prevent dirty data from getting into the database (before insert/update/delete) – (\* This is limited \*)**

type checking

integrity constraints

use of triggers

**(2) Clean the dirty data that got through – (\* It takes a lot of work \*)**

Using data cleaning tools, if available

Manual cleaning (using a data browser tool)

**4.2.1.4Outliers**

**Ex) 평균을 구하려는데 200키로가 넘는사람**

**에러일 가능성이 높다 – 삭제**

**에러는 아닌데 중요하지 않다 – 가중치를 낮게**

**데이터셋.decribe() - Get Statistics on the Data**

**데이터셋.info() - Show Data Types for the Features**

**데이터셋.columns.values - List Feature Names**

**데이터셋.drop([어떤놈들,axis=숫자])**

**데이터셋[[‘가져올세로1’,'가져올세로2']].groupby(['세로'], as\_index=False).mean().sort\_values(by='세로', ascending=False)**

**4.2.2Text preprocessing**

**noise removal**

Remove text file headers, footers.

Remove markup and metadata from HTML, XML, JSON, etc. documents

**tokenization/segmentation/lexical analysis**

**.이나 ‘ , - 와 같은 것들을 지워주는 것**

**Normalization**

word stemming – ing나 ed를현재형으로 바꿔주는 것

word lemmatization – finest -> fine 처럼 바꿔주는 것

everything else

**4.2.3Data discretization**

**Sort data and partition into bins**

데이터를 bin으로 나누어 정렬

**Data in each bin may be smoothed by bin means, bin median, or bin boundaries**

mean이나 median과 같은 값들로 smoothing 하는것

**4.2.4Data normalization/standardization**

Data scaling means either applying a transformation to make the data roughly normally distributed, or scaling data to fall within a small, specified range

데이터 확장이란 변환을 적용하여 데이터를 대략적으로 정규 분포를 따르거나, 데이터를 작은 지정된 범위에 포함하도록 확장하는 것을 의미합니다.

**Why Scale (Normalize) Data?**

**머신러닝 알고리즘은 숫자의 크기에 비례해서 가중치를 둔다**

**머신러닝 알고리즘은 피쳐수가 적어질수록 성능이 좋아진다.**

**카테고리컬 데이터는 스케일링 할 필요 없음**

**Data Scaling**

**min-max scaling**

**데이터의 값들을 0~1 사이의 값으로 변환시키는 것**

**각 변수가 정규분포(bell-shape)가 아니거나 표준 편차가 작을 때 효과적**

**Standard Scaler와 같이 이상치 존재에 민감**

**z-score (standard) scaling**

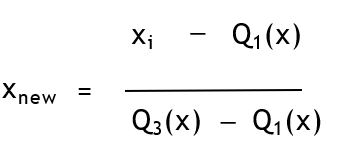
**데이터의 최소 최대를 모를 때 사용**

**이상치가 많다면 사용하지 않는 것이 좋음 .**

**Ex) low entropy (less random data)**

**Ex) near normally distributed**

**robust scaling**

****

**Standard scaler에 비해 이상치의 영향이 적어짐**

**decimal scaling**

**가장 큰 숫자의 개수를 세서 10^개수 로 나누는 것**

**log normalization**

**값이 여러 규모에 걸쳐 있는 경우에 사용됩니다.**

**Ex)** **large range of data: (i.e. $4 to $120,000,000),**

**Ex)** **skewed data (often large range)**

**vector normalization**

**벡터 정규화기(방향 코사인, 좌표 정규화기)는 n개의 피쳐 수에 대해 n차원 공간의 크기로 각 값을 분할하여 스케일링합니다.**

**4.2.5Encoding for data mining algorithms**

**4.3Feature Engineering**

**4.3.1Feature Creation**

**4.3.1.1 Deriving Features from Existing Features**

**주어진 피쳐를 기반으로 새로운 피쳐를 만드는것**

**Terminology**

**entity –** Pandas Data Frame, RDB table

**entity set** – set of entities, RDB tables

**shared variable** – RDB foreign key

**transformation –** deriving a new feature from existing features within an entity

**aggregation** – merging different entities, or calculating statistics on a group of rows in an entity and merging the result as a feature of another entity

**es = ft.EntitySet(id = 'clients')**

**es = es.entity\_from\_dataframe(entity\_id = 'clients', dataframe = clients, index = 'client\_id', time\_index = 'joined')**

**Add an Entity to an Entity Set**

**es = es.entity\_from\_dataframe(entity\_id = 'payments', dataframe = payments,**

**variable\_types =**

**{'missed':ft.variable\_types.Categorical}, make\_index = True, index = 'payment\_id', time\_index = 'payment\_date')**

**# Group loans by client id and calculate mean, max, min of loans**

**stats = loans.groupby('client\_id')['loan\_amount'].agg(['mean', 'max', 'min'])**

**# Merge with the clients dataframe**

**stats = clients.merge(stats, left\_on = 'client\_id', right\_index=True, how = 'left')**

**# Create a relationship between clients and loans**

**r\_client\_previous = ft.Relationship(es['clients']['client\_id'],**

**es['loans']['client\_id'])**

**# Add the relationship to the entity set**

**es = es.add\_relationship(r\_client\_previous)**

**# Relationship between previous loans and previous payments**

**r\_payments = ft.Relationship(es['loans']['loan\_id'],**

**es['payments']['loan\_id'])**

**# Add the relationship to the entity set**

**es = es.add\_relationship(r\_payments)**

**4.3.1.2 Converting Values to New Features**

**두개의 연관이 있는 엔티티를 기반으로 새로운 피쳐를 만들어서 새로운 피쳐에 추가하는것**

**One-Hot Encoding**

**4.3.2Feature Selection**

**디멘션이 커지면 시간이 증가하고 리소스가 엄청 필요하게 되는데 이는 결과를 가시화하는데 어려움을 초래함**

**Filter Methods – 최적의 피쳐 서브셋을 찾는 것**

**Wrapper Methods – 데이터 마이닝을 사용해서 찾는 것**

**Embedded Methods – 머신러닝 알고리즘을 사용해서 찾는 것**

**4.3.2.1Univariate selection SelectKBest**

**#apply SelectKBest class to extract top 10 best features**

**bestfeatures = SelectKBest(score\_func=chi2, k=10)**

**fit = bestfeatures.fit(X,y)**

**dfcolumns=pd.DataFrame(X.columns)**

**dfscores= pd.DataFrame(fit.scores\_)**

**#concatenate two dataframes for better visualization**

**featureScores = pd.concat([dfcolumns, dfscores],axis=1)**

**featureScores.columns = ['Specs',‘ Score']**

**#name the dataframe columns**

**print(featureScores.nlargest(10,'Score'))**

**#print 10 best features**

**4.3.2.2Feature importance scoring ExtraTreesClassifier**

**from sklearn.ensemble import ExtraTreesClassifier**

**model = ExtraTreesClassifier()**

**model.fit(X,y)**

**print(model.feature\_importances\_)**

**#use built-in class feature\_importances of tree-based classifiers**

**#plot graph of feature importances for better visualization**

**feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns)**

**feat\_importances.nlargest(10).plot(kind='barh')**

**4.3.2.3Correlation matrix with heatmap seaborn**

**data = pd.read\_csv("D://Blogs//train.csv")**

**X = data.iloc[:,0:20] #independent columns**

**y = data.iloc[:,-1] #target column, i.e., price range**

**#get correlations of pairs of features in the dataset**

**corrmat = data.corr()**

**#corr() computes pairwise**

**correlations of features in a Data Frame**

**top\_corr\_features = corrmat.index**

**plt.figure(figsize=(20,20))**

**#plot the heat map**

**g=sns.heatmap(data[top\_corr\_features].corr(),annot= True,cmap="RdYlGn")**

**4.3.3Feature Reduction – 데이터를 바꾸는 것**

**4.3.3.1Principal Component Analysis (PCA)**

**많은 정보가 소수의 피쳐에 집중되도록 데이터를 변환하는 것**

**-> 머신러닝 속도 증가, 데이터 가시화**

**# Standardize the features 정규화**

**x = StandardScaler().fit\_transform(x)**

**pca = PCA(n\_components=2) 정규화한 데이터를 pca에 적용**

**principalComponents = pca.fit\_transform(x) principalDf = pd.DataFrame(data =**

**principalComponents, columns =**

**['principal component 1', 'principal component 2'])**

**pca.explained\_variance\_ratio\_**

**사용하면 유지하는 배리언스의 비율을 알 수 있음**

**Scree plot을 사용해서 elbow 지점이 유지하는 variance 개수임**

**Pca의 한계점**

**데이터를 흐뜨려놓는게 목적인데 이게 적절하지 않을 수 있음**

**합쳐진 베리언스를 이해하기 어려울 수 있음**

**직각을 유지하기 어려울 수 있음**

**4.4Data Reduction**

**4.4.1Feature Selection/Reduction**

**4.4.2Data Filtering**

Ex) “below age 30

**4.4.3Data Summarization**

1.실제 데이터를 사용하지 않고 데이터에 대한 요약 정보만 사용하는것

2.데이터셋을 몇 개의 그룹으로 나누었을경우 데이터분석에는 오리지널 데이터셋은 버리고 각 그룹에 대한 요약 정보만 사용하는것

3.데이터를 히스토그램(빈)으로 정의하고 히스토그램의 각 빈에대한 요약 정보만 사용하는것

**4.4.4Concept Hierarchy**

하층의 자세한 데이터대신 상층의 요약된 데이터만 사용하는것임

하층의 임직원의 데이터 대신 상위 계층을 사용해도 괜찮은 경우

**4.4.5Data Compression**

**Lossless Compression 복구가능**

Original data can be reconstructed

**Lossy Compression 복구 불가능**

Only an approximation of the original data can be constructed

Wavelet transforms, and principal component analysis (**PCA**)

**4.4.6Numerosity Reduction**

**4.4.6.1Parametric Methods**

Use a model to fit data, and store only the model parameters (not the original data). 모델을 사용해서 모델을 핏하고 오리지널 대신 모델에 사용되는 파라미터만 사용하는 것

**4.4.6.2NON-Parametric Methods**

**4.4.6.2.1Discretization**

Numerical 배리어블의 range를 제한된 숫자의 인터벌(빈)로 나누고 각 인터벌과 숫자들에 대한 정보를 저장하는 것

**Equal width bins**

**간단하지만 의미없는 아웃라이어에 빈을 낭비할 수도 있음**

**Equal frequency bins**

**데이터가 많을경우나 분포가 한쪽으로 쏠려있을 때 사용**

**머신러닝에서는 카데고리컬 데이터가 문제를 야기하는데 이거에도 맞지 않을 수 있음.**

**Entropy based**

**4.4.6.2.2Clustering**

**데이터를 그루핑해서 크러스터별로 나눠진것**

**4.4.6.2.3Sampling**

**Techniques**

**Simple random sampling**– 전체 데이터셋에서 여러 차례 랜덤하게 데이터를 뽑아서 모으는것 , 한번 뽑은 데이터는 다시 뽑을수도 있고 안뽑을 수도 있음

**Systematic sampling** – 전체 데이터셋에서 매 k번째의 데이터를 보는것

**Statified random** – 전체 파퓰레이션을 몇 개의 그룹으로 나눠놓고 각 그룹에서 랜덤하게 데이터를 뽑느것 – 샘플링할때 치우치지 않고 데이터를 뽑을 수 있음

**Probability-Proportional-to-Size random sampling** – 사이즈를 서로 다른 여러 그룹으로 나누는것 각 그룹에서 그 사이즈에 비례에서 랜덤하게 데이터를 뽑는것

**5.Data Analysis**

**6.Evaluation**

**7.Deployment**

**1.Correlation**

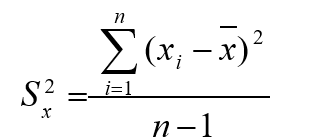
통계에서 쓰이는 기법으로 변수사이에 상관관계정도를 구하기 위한 기법

Correlation이 있다고해서 인과관계가 항상 있다고 생각하면 X

**Linear Regression Line**

**Scatter Plots**

**Variance**

****

**Covariance**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**X 가 커지는데 y도 커지면 cov(x,y)=pos**

**X 가 커지는데 y는 작아지면 cov(x,y)=neg**

**아무 관계가 없으면 cov(x,y)=0**

**문제점**

**변동에 대한 양이기 때문에 기본적으로 데이터가 가지고 있는 표준편차, 분산에 영향을 준다**

**Variance가 크다면 covariance가 클 확률도 높다.**

**Pearson’s r 각각의 변수에 무관하게 두 변수의 관계에만 영향을 받으면 어떨까 해서 나온것**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**문제점**

**R이 1이나 -1일 경우 완전한 상관관계가 있다는 것인데**

**예외적인 데이터값 때문에 r이 바뀜**

**Spearman’s Correlation Coefficient *(rs) 예외적인 데이터 때문에 r이 바뀌는 것 때문에 나온 것***

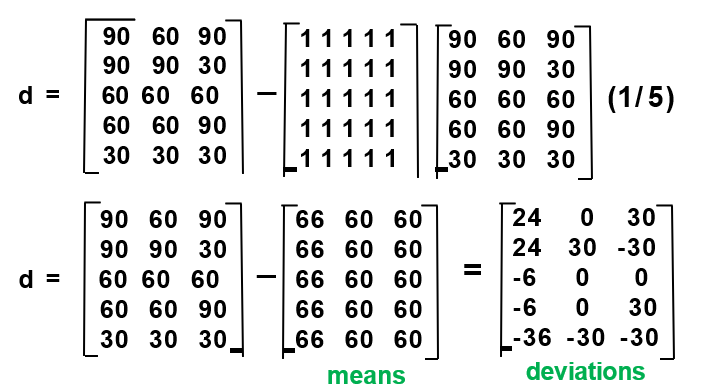
**텍스트이(가) 표시된 사진

자동 생성된 설명**

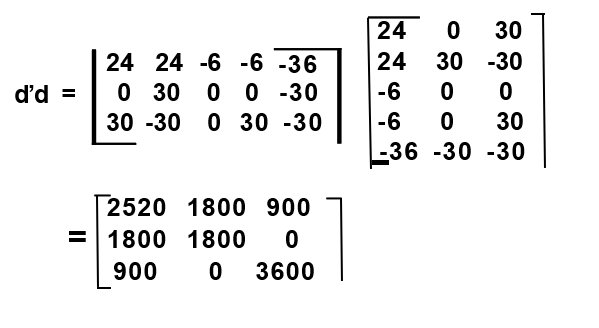
**V 매트릭스**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | **Math** | **English** | **Art** |
| **1** | **90** | **60** | **90** |
| **2** | **90** | **90** | **30** |
| **3** | **60** | **60** | **60** |
| **4** | **60** | **60** | **90** |
| **5** | **30** | **30** | **30** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **90** | **60** | **90** |  |
| **90** | **90** | **30** |  |
| **M =** | **60** | **60** | **60** |  |
|  | **60** | **60** | **90** |  |
|  | **30** | **30** | **30** |  |

****

**V = d‘d ( 1 / n )**

****

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**2.Regression**

**2.1 Linear Regression**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**SSE**

**에러 제곱의 합 - SSE가 작을 수록 좋은 fit임**

**SSE 실제 데이터가 예측값과 에러의 합으로 표현되는데 예측을 하고 난 뒤에도 남는 에러**

**Regression으로 셜멍 X**

****

**SSR**

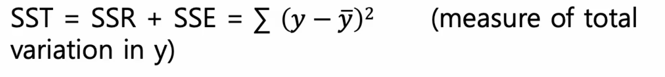
**개별 데이터의 차이가 아니라 y평균 값 과의 차이를 제곱하고 합**

**SSR 기울기가 가파르다 – x의 범위에 대해서 y가 퍼져잇음 이는 직선의 방적식으로 설명이가능**

**Regression으로 셜멍 가능**

****

**SST**

****

****

**SSE가 0 이라는 것은 모든게 선위에 놓여있다.**

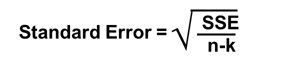
**SSE가 SST값에 가까울때 R^2이 0에 가까워질것임**

****

**현재 있는 input 외에 추가적인 input을 가져와서 예측을 하면 감소하진 않지만 R^2이 조금이라도 증가한다.**

**R^2을 증가시키는 것은 좋은것은 아니다**

**Predict를 들리는것은 overfitting을 발생시킬 수 있음 성능을 떨어뜨릴수있다는것임**

** where n is the number of observations and**

**k is the total number of variables in the model**

**2.2 Polynomial Regression**

**M을 증가시킬수록 R^2는 계속 증가함 감소하지는 않음**

**R^2 adj 는 증가하다가 감소하는 구간이 생기는데 그구간을 선택**

**텍스트, 시계이(가) 표시된 사진

자동 생성된 설명**

**2.3 Multiple Regression**