Part 1: feature engineering

Exercise 1:

Consider the following dataset containing information about the weather and the number of people who visited a park on a given day:

Temperature	Cloud Cover	Humidity	Weather
75	Sunny	Low	Sunny
80	Partly Cloudy	High	Sunny
85	Overcast	High	Rainy
70	Sunny	Medium	Sunny
65	Overcast	Medium	Stormy
60	Partly Cloudy	Low	Sunny
90	Overcast	High	Rainy

Your task is to select the most relevant features (weather conditions) to predict the number of visitors to the park. To do this, you will calculate the information gain for each feature.

1. Discretize the feature Temperature; so we have three levels of temperature: warm, hot, and very hot.
2. Calculate the Entropy of the target class Weather, using the formula:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot \log_2(P(x_i))$$

Where, P(xi) is the appearance probability of value x_i among the values of the feature X, and n is the number of records of X.

3. Calculate the information gain of each feature in regard to the target feature Weather, using the information gain formula:

$$IG(A, X) = H(X) - \sum_{i=1}^{n} P(x_i)\, H(A|x_i),$$

Where H(A |xi) is the entropy of the feature A calculated on the portion of data where the target feature has the value x_i.

4. Order the features according to their IG.

Exercise 2:

Consider the following dataset composed of 6 data rows and 3 variables.

	x	y	z
R1	12	24	6
R2	17	15.5	-2
R3	12	13	3
R4	6	13.5	-2.5
R5	17	21	7.2
R6	4	20.3	-0.9

Your task is to extract a reduced set of features using PCA method. PCA provides a matrix of eigenvectors that explains the variance of the original variables. To reduce data using PCA, we run the next operation:

$$R = X V$$

Where, R is the reduced data (R has the shape of r data rows, and m data columns), X is the original data matrix (i.e., it has the shape of r data rows, and n columns). V is the eigenvectors matrix (it has the shape of n data rows, and m data columns).

To calculate the reduction matrix V, we follow the next steps:

1. Calculate the mean and standard deviation for each variable (X, *Y*, *Z*).
2. Standardize the data using the Z-score to get the standardized matrix S. Next, the z-score formula.

$$x=(x\text{-}Mean(x))/STD(x),$$

Where, x is a feature vector, and STD(x) is the standard deviation of x.

3. Calculate the covariance matrix, C, (i.e., C has the shape of (n ,n).) of the standardized data. The covariance matrix is given by:

$$C(x,y) = Cov(x,y) = x\, y/(n-1)$$

x and y are two feature vectors, and n is the number of elements in the data vectors (i.e., number of variables).

4. Find the eigenvalues $(\lambda 1, \lambda 2, \lambda 3)$ and corresponding eigenvectors $(\boldsymbol{v}1, \boldsymbol{v}2, \boldsymbol{v}3)$ of the covariance matrix by solving the next equation:

$$det(C\text{-}D)=0$$

Where, D is a diagonal matrix having in its diagonal $(\lambda 1, \lambda 2, \lambda 3 \ldots)$ values.

Also, to get the eigenvectors corresponding to the given eigenvalue λi, we solve the next linear equation:

$$(C - \lambda iI)\, vi = 0$$

Where, I is the identity matrix of shape (n,n), and vi is the eigenvector of shape (1, n) corresponding to the eigenvalue λi.

5. Sort the eigenvalues in descending order and arrange the corresponding eigenvectors accordingly.
6. Decide how many principal components to keep based on the explained variance (i.e., each eigenvalue explains a portion of variance.). You can decide to keep, for example, the first two principal components if their sum represents 90% of the variance (sum of their eigenvalues divided on the total sum is bigger than 0.9).
7. Form the projection matrix V by only keeping eigenvectors corresponding to the highest eigenvalues.
8. Calculate the reduced data matrix R by projecting the standardized data X into the selected principal components to obtain the reduced-dimensional representation using.

Practical work:

Feature selection

Consider the Titanic dataset for your practical work, and test feature-engineering impact on the results.

- Get the data, and then give a short description using corresponding statistical indicators.
- Clean the data by dealing with missing values, dropping unnecessary features, standardizing the data…etc.
- Measure the information gain of each feature on the Survived passengers.
- Test different configurations to build a classification model using K-NN algorithm by: (a) testing different number of features, and (b) different number of K. (c) different similarity measure.
    - Use precision for validating results.
- Visualize and interpret results.

Feature extraction

Let's consider the Iris flower data set that contains 150 data instances, and 4 data features.

- Get the data, and then give a short description using corresponding statistical indicators.
- Clean the data by dealing with missing values, dropping unnecessary features, standardizing the data…etc.
- Extract new featuring using PCA, and rank them according to their variance.
- Test different configurations to build a clustering model using K-Means algorithm by: (a) testing different number of features, and (b) different number of Clusters. (c) different similarity measures.
    - Use a density based technique to examine your model,
    - Examine the size of the generate clusters.
- Visualize and interpret results.

Part 2: basic recommender system.

We try to build a basic recommender system using the singular value decomposition technique SVD. MovieLens 100K is a public dataset that has 943 users, 1683 movies, and around 100K ratings given by users to movers in a range of 1 to 5. Each rating is associated with a timestamp as shown in the next sample.

Table 1 Example of MovieLens dataset structure with 2 users and 3 movies.

User_id	Movie_id	Rating	Timestamp
1	1	4	455664
1	2	1	455555
2	2	4	444555
2	3	2	555554

A main challenge in building a reliable recommender system is data sparsity, which represents the portion of missing values that limits the performances of any statistical technique. The previous table can be turned into 2D table (i.e., rows represents users, columns represents movies.) to show the sparsity of data, where X represents a missing value or a target value that we like to predict.

Table 2 Structure of dense table with missing values.

1	2	3
4	1	X
X	4	2

The application of SVD on MovieLens 100K generates three matrixes U, S, and V. Where, U represents a matrix of |U| vector represents the hidden preferences of a given user u from U to the set of movies. Similarly, V represents a matrix of |V| vectors each of which explains how a given movie v from M is liked by users.

To apply and test SVD:

1. Load MovieLens 100k **data**, and omit timestamp column. Data takes the form lines, separated by a token (i.e., comma, or a tabular.) such as a CSV file.
2. Split the **data** into **train** and **test** parts by considering 80% and 20% for the splits respectively.
3. Convert the **train, and test** into dense tables named **training_data, and testing_data** (i.e., as in the previous example, we pass from table 1 to table 2).
4. Calculate the **global_mean** of all ratings in the training data.
5. Fix the next controlling parameters:
   - **Lamp**: a normalization parameter that we set equal to 0.99.
6. Calculate the users' bias **bu** of each user **u** calculated as:

$$b_u = \frac{\sum_{i \in I_u} (rating(u,i) - global\_mean)}{lamb + |I_u|}, I_u \text{ is the set of items rated by u.}$$

Ratings			Users' Bias
1	2	3	?
4	1	X	?
X	4	2	?

7. Calculate the bias **bi** of each item calculated as:

$$b_i = \frac{\sum_{u \in U_i} (rating(u,i) - global\_mean)}{lamb + |U_i|}, U_i \text{ is the set of users who rated i.}$$

rating			
	1	2	3
	4	1	X
	X	4	2
Items bias	?	?	?

8. Fill the missing values of the **traing_data** using the formula:

$$missing(u,i) = b_u + b_i + global\_mean$$

1	2	3
4	1	?
?	4	2

9. Apply SVD on the **training_data** and get **U, S, V**.

(U,S,V) = SVD(training_data)

10. Reduce U, S, V to keep only **Approx** column for **U**, and **V**, and **Approx** columns and row for **S**.
    - U = U[:, Approx.]
    - V = V[Approx, :]
    - S = S[Approx; Approx]

11. Calculate **Z** as:

$$Z = U \cdot S \cdot V$$

12. Calculate the **MAE** of the model using the next formula:

$$MAE = \sum_{(u,i) \in (U,I)} |testing\_data(u,i) - Z(u,i)|, where \quad testing\_data(u,i) \neq 0$$

13. Redo the steps 9 to 12 by setting **Approx** equal to [5,10,15,20,25,30,35,40,45,50]
14. Plot a bar graph representing the **MAE** for each configuration.