JMAN GROUP

Feature Engineering

jmangroup.com

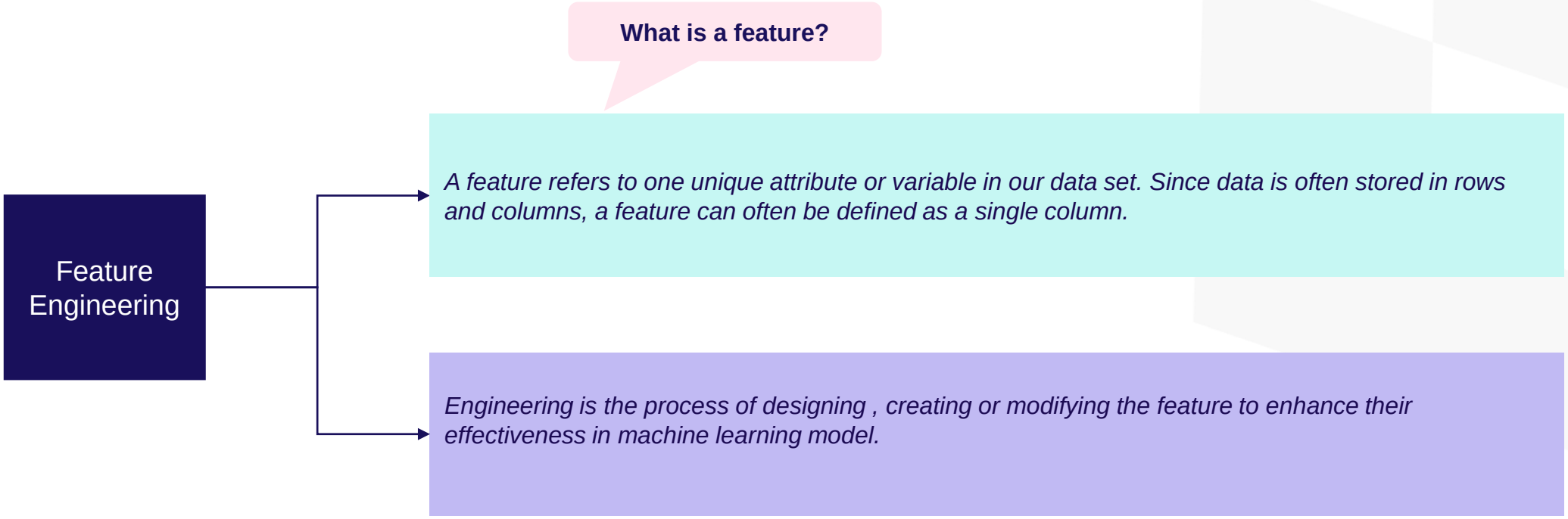
FT FINANCIAL TIMES
statista
UK'S LEADING MANAGEMENT CONSULTANTS 2022

Context

Feature Engineering: Introduction

More data can beat clever algorithms, but better data beats more data

Feature engineering is the process of **selecting, transforming, or creating data attributes (features) to enhance the performance of machine learning models**, allowing them to better understand and make predictions from raw data.



Features for recognizing a chair?

A little brainstorming on what could be the features for identifying a chair











Context

1 Introduction

2 Feature Creation

3 Feature Bucketing

4 Feature Transformation

5 Feature Correlation

6 Feature Selection

Feature Creation: Enhancing Data for Machine Learning

Creating features for getting better insights

Feature creation involves transforming raw data into new informative attributes that enhance a machine learning model's ability to understand patterns and make accurate predictions.

We are working with a dataset for predicting customer churn for a subscription-based service. The dataset includes the following columns:

CustomerID	A unique identifier for each customer
MonthsActive	The number of months the customer has been active
TotalSpent	The total amount of money the customer has spent
SupportTickets	The number of customer support tickets raised
EmailsSent	The number of marketing emails sent to the customer.
SubscriptionType	The type of subscription (e.g., Basic, Premium).

Let's create new features with the help of existing data to improve the accuracy of your churn prediction model.

Feature	Importance
AverageSpendingperMonth	This feature represents the average spending per month by each. (TotalSpent / MonthsActive)
Support Ticket Rate	This feature can indicate how often customers contact support relative to their duration as subscribers. High support ticket rates might be an indicator of potential churn.
Engagement Score	A composite feature that combines MonthsActive and EmailsSent. For instance, multiply the number of months by the number of emails sent. Customers with higher engagement scores may be less likely to churn.
Tenure Category	Convert MonthsActive into categories such as "New," "Intermediate," and "Long-term." This feature can help capture the customer's loyalty or stage in their subscription, which may affect churn.

Outcome - By adding these new features to the dataset, we can provide the machine learning model with more relevant information, potentially improving its ability to predict customer churn accurately. Feature creation helps transform the raw data into a more informative representation of the problem at hand and can significantly impact the model's performance.

Why do we need feature creation?

The purpose of feature creation is to **enhance model performance by capturing intricate patterns, addressing non-linearity, and converting categorical data**. It serves to improve model interpretability, manage missing information, and ultimately contribute to the development of more robust and effective machine-learning models.

Examples of feature creation-

We have data on the prices of properties in a city which shows the area of the house and the total price.

Sq. Ft	Amount	Places
2400	9 M	chennai
3200	15 M	pune
2500	10 M	chennai
2500	8.9 M	Mumbai
2100	1.5 M	Mumbai

The data might not be giving us a correct picture, to identify it, we will add one extra column 'Cost Per Sq. Ft'

Sq. Ft	Amount	Cost Per Sq. Ft	Categorical values of places
2400	9 M	4150	1
3200	15 M	4944	2
2500	10 M	3950	1
2500	8.9 M	3600	3
2100	1.5 M	510	3

The new column 'cost per sq. ft' clearly indicates us the last value is not correct.

Based on the above example, it becomes evident that generating a new feature allows us to gain accurate insights into the validation of the input data.

Context

1 Introduction

2 Feature Creation

3 Feature Bucketing

4 Feature Transformation

5 Feature Correlation

6 Feature Selection

Categorizing features in the data provides better insights and reduces the number of unnecessary features

Feature bucketing is a data preprocessing technique that involves grouping numerical or continuous data into discrete intervals or “buckets“. These buckets represent different ranges or categories of values, making it easier to analyze and model the data. Feature bucketing can help capture non-linear relationships, reduce the impact of outliers, and simplify the interpretation of models by converting continuous data into categorical representations.

Suppose we are working with a dataset that includes information about individuals, and one of the features is "Age." Now we want to create age groups for analysis or modeling.

Name	Age
Frey	30
AJ	42
Sathy	55
Deep	18

We decided to create age groups to simplify the analysis. The defined age bins:

Age	Group
0-18	Child
19-35	Young Adult
36-55	Adult
56-100	Senior

Name	Age	AgeGroup
Frey	30	Young Adult
AJ	42	Adult
Sathy	55	Adult
Deep	19	Child

Context

1 Introduction

2 Feature Creation

3 Feature Bucketing

4 Feature Transformation

5 Feature Correlation

6 Feature Selection

Without good data quality, high-performance algorithms become ineffective

Feature transformation is the process of modifying or converting features in a dataset to create new representations that enhance the effectiveness of machine learning algorithms. This transformation can involve changing the scale, distribution, null handling, or structure of features to make them more suitable for modeling.

Why do we need feature transformation?

- Normalization of data like scaling features to a common scale, prevents certain features from dominating others.
- **Normal Distribution** is a very important distribution in Statistics, which is key to many statisticians for solving problems in statistics. More likely, features in the real-life data are not normally distributed and will follow a **skewed distribution.**
- By applying methods, such as logarithmic or square root transformations, can help normalize skewed feature distributions.
- This is crucial for improving the performance of models that assume normality, like linear regression.
- Some machine learning models, like linear regression, assume a linear relationship between features and the target variable.
- Feature transformation helps establish or enhance linearity, improving the model's accuracy and interpretability which leads to more stable and reliable model predictions, particularly in cases where the data violates assumptions of normality and homoscedasticity.

Different types of transformation techniques

1. Function Transformers
    - Log Transform
    - Square Transform
    - Square Root Transform
    - Reciprocal Transform
    - Custom Transform
2. Power Transformers
    - Box-Cox Transform
    - Yeo-Johnson Transform
3. Quantile Transformers

Examples for transformation steps

Example of power transformation -

The Power Transformer automates this decision-making by introducing a parameter called **lambda**. It decides on a generalized power transform by finding the best value of lambda using either the –

1. **Box-Cox transform** OR 2. **The Yeo-Johnson transform**

lambda (λ) is a parameter that determines the type and strength of the transformation applied to the data

Since all the input values are positive, we can use the Box-Cox transform

Code -

```
from sklearn.processing import PowerTransformer

Scaler = PowerTransformer(method = 'box-cox')
df[column_names] = Scaler.fit_transform(features.values)
```

Income	Age	Department
15000	25	HR
1800	18	Legal
120000	42	Marketing
10000	51	Management

Income	Age	Department
0.125158	-0.597385	HR
-1.395497	-1.301984	Legal
1.419403	0.681202	Marketing
-0.149064	1.218168	Management

Context

1 Introduction

2 Feature Creation

3 Feature Bucketing

4 Feature Transformation

5 Feature Correlation

6 Feature Selection

Feature Correlation: Understanding the significance of feature correlation

Exploring feature dependencies and its correlation

Feature correlation is a method **to understand the relationship between two or more features in your dataset,** This analysis offers valuable insights into how changes in one feature are linked to changes in another. High correlation may indicate redundancy or a potential predictive relationship, it helps guiding decisions related to feature selection, model interpretability etc.

Significance of Feature Correlation

- **Prediction Power**: Correlation facilitates the prediction of one attribute from another, enhancing the overall predictive capabilities of your models.
- **Causal Inference**: It can sometimes unveil causal relationships, shedding light on the underlying dynamics within your dataset.
- **Modeling Foundation**: Many modeling techniques rely on correlation as a foundational element, emphasizing its role in constructing reliable and accurate models.
- **Bias Reduction**: Feature correlation plays a pivotal role in minimizing biases present in the data, ensuring a more balanced and representative analysis.

Considerations for Feature Correlation Thresholds

- Evaluate how feature correlations may affect the performance of models.
- Identify and address redundant features that provide similar information.
- High correlations (100%) may introduce bias, emphasizing the need for thoughtful threshold selection.
- Using visualizations like heatmaps to intuitively understand the correlation structure of features.
- A correlation coefficient (r) close to 0 indicates a weak or no linear relationship.
- Correlation coefficients between +0.3 and +0.7 (or between -0.3 and -0.7) suggest a moderate relationship.
- A widely used threshold is 0.75 (75%).

Context

1 Introduction

2 Feature Creation

3 Feature Bucketing

4 Feature Transformation

5 Feature Correlation

6 Feature Selection

Feature selections helps in reducing the noise in data and time take to train a model

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

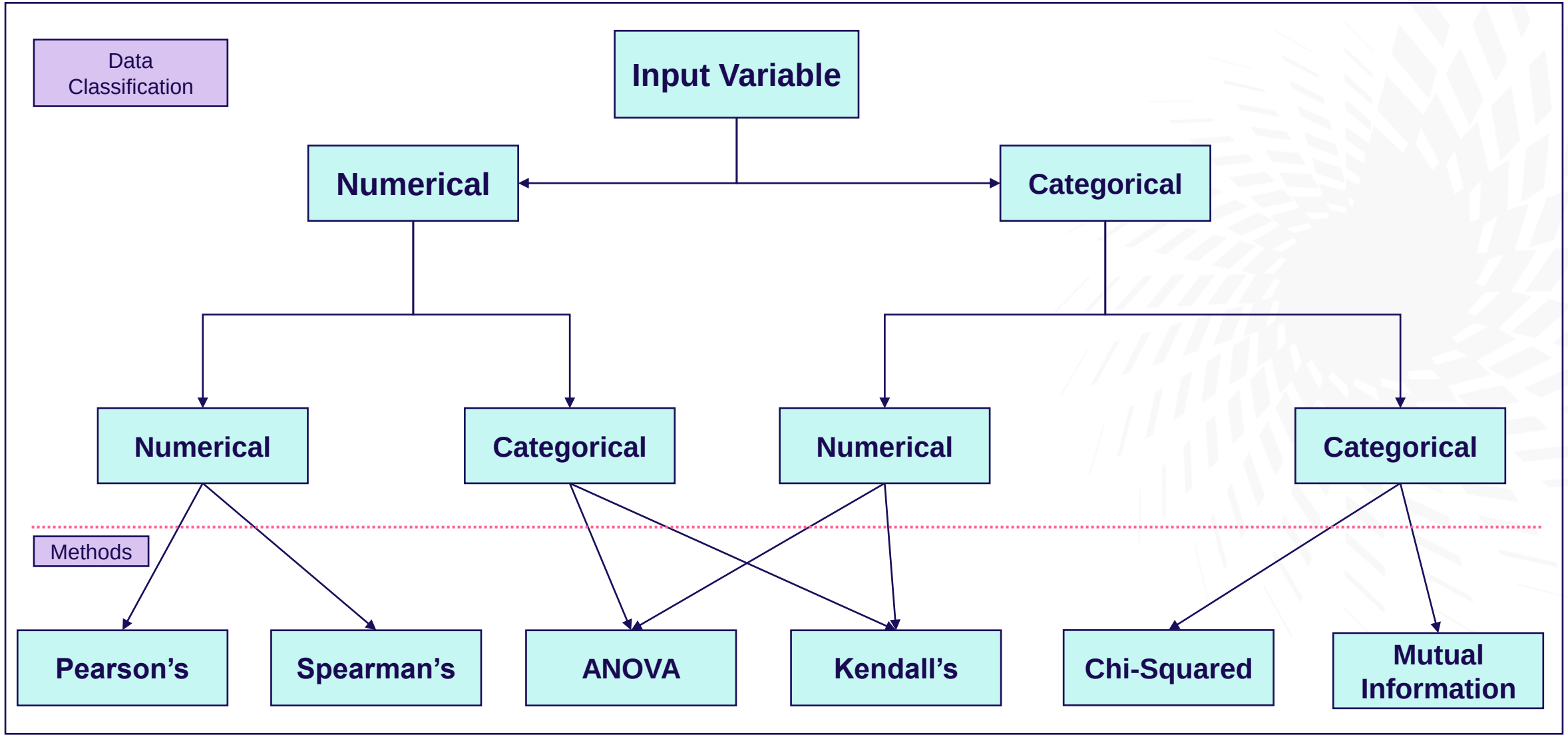
Why do we need feature selection?

- In the machine learning process, feature selection is used to make the process more accurate.
- Feature selection can enhance model performance by focusing on the most relevant features, reducing noise and overfitting.
- A smaller set of features makes it easier to interpret and communicate the model's insights.
- Removing irrelevant or redundant features helps eliminate noise in the data.
- Noise can negatively impact model performance and make it harder to discern meaningful patterns.
- Feature selection mitigates multicollinearity issues where two or more features are highly correlated.

Different types of feature selection

1. Wrapper methods
    - Forward Selection
    - Backward Elimination
    - Recursive Feature Elimination
2. Filter methods
    - Pearson’s correlation
    - LDA
    - ANOVA
    - Chi-square
3. Embedded methods
    - L1 regularization
    - L2 regularization

Feature Selection Methods



Feature selection can be performed by using predefined algorithms

Below are few examples of feature selection methods

1 Pearson Correlation method

- The **Pearson correlation method** basically comes into place only when we need to perform a feature selection between numerical data type features.
- In other terms we can say that it is the most common way of measuring a linear correlation between numerical features.
- There are two types of correlations.
  - **Positive Correlation:** means that if feature A increases, then feature B also increases or if feature A decreases, then feature B also decreases. Both features move in tandem, and they have a linear relationship.
  - **Negative Correlation:** means that if feature A increases, then feature B decreases and vice versa

2 Anova method

- **ANOVA i.e.(Analysis of Variance)** is a statistical method used to assess the variability between the means of two or more groups.
- **Purpose of ANOVA in Feature Selection:** ANOVA is commonly used in feature selection to identify features that exhibit significant variance across different categories, aiding in the discrimination between these groups.
- **ANOVA** test is performed when we need to do a feature selection process between numerical and categorical features.
- The test produces an F-statistic and a **p-value**, where a low p-value indicates that there are significant differences among the means of the groups, making a feature more likely to be relevant in the selection process.

3 Chi-squared method

- A **Pearson's chi-square test** is a statistical test for categorical data. It is used to determine whether your data are significantly different from what you expected.
- The test produces a **p-value**, indicating the probability of obtaining the observed results if there is no true association between the variables. A low p-value (typically below a significance threshold like 0.05) suggests a significant correlation.

Feature Correlation Matrix

Pearson’s Correlation matrix -

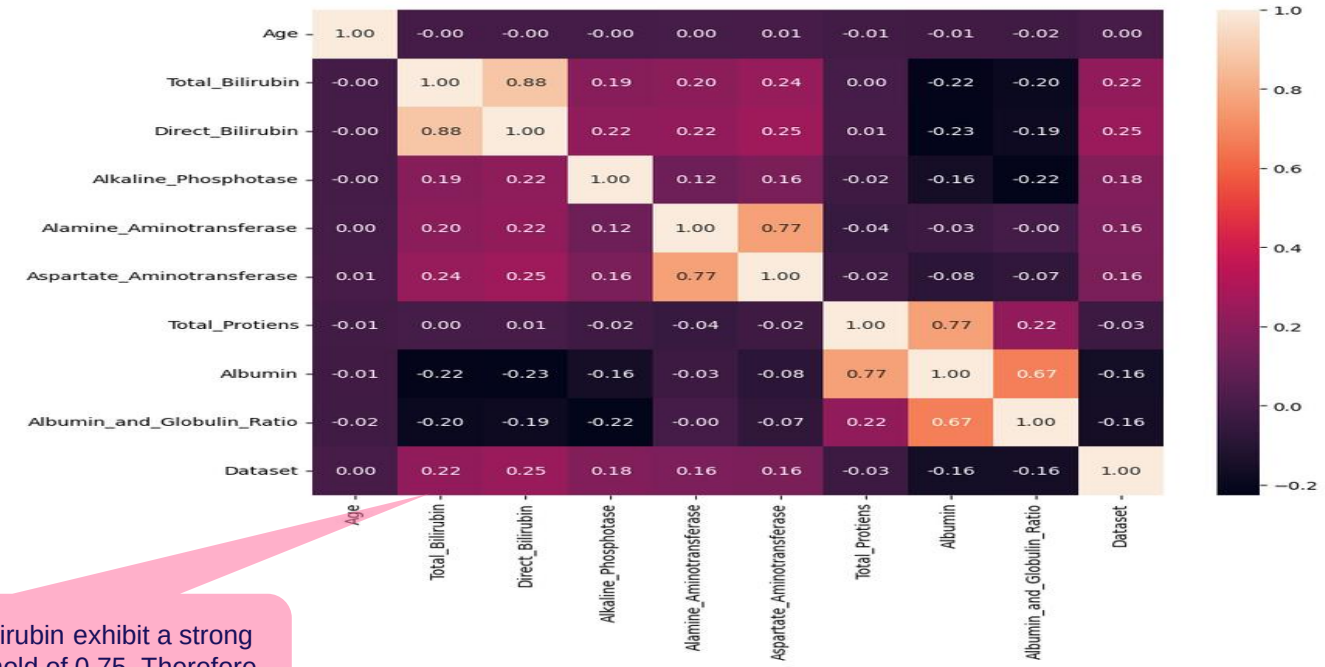
Sample code for visualizing **Pearson’s correlation matrix**

Selecting features using Correlation

```
# Correlation matrix
correlation = num_df.corr()

#Plotting Heatmap
plt.figure(figsize = (15,6))
sns.heatmap(correlation, annot = True)
```

Heatmap provides a better understanding of features’ relation to each other as well as the target variable



Total_Bilirubin and Direct_Bilirubin exhibit a strong correlation exceeding a threshold of 0.75. Therefore, we can retain only one of these features

Feature Engineering Examples

29 November 2023

Exercise Questions

1. Can you think of a new feature that might provide valuable information for predicting churn?
2. How might you transform it to make the data more suitable for machine learning models?
3. How do you handle situations where a categorical feature has a large number of unique values?
4. How will you bucket the feature to simplify its representation?

Step 1: Creating New Features

We've calculated the percentage of daytime, evening, and night minutes, summed up total calls, charges, and introduced interactions between variables like having an international plan and voicemail plan, providing a richer understanding of customer behavior for more effective analysis.

```
df['Daytime Minutes Percentage'] = (df['Total day minutes'] /
                                   (df['Total day minutes'] + df['Total eve minutes'] + df['Total night minutes'])) * 100
df['Evening Minutes Percentage'] = (df['Total eve minutes'] /
                                   (df['Total day minutes'] + df['Total eve minutes'] + df['Total night minutes'])) * 100
df['Night Minutes Percentage'] = (df['Total night minutes'] /
                                 (df['Total day minutes'] + df['Total eve minutes'] + df['Total night minutes'])) * 100
df['Total Calls'] = df['Total day calls'] + df['Total eve calls'] + df['Total night calls']
df['Total Charges'] = df['Total day charge'] + df['Total eve charge'] + df['Total night charge'] + df['Total intl charge']
df['Intl Plan Interaction'] = df['International plan'] * df['Total intl minutes']
df['Voicemail Plan Interaction'] = df['Voice mail plan'] * df['Number vmail messages']
df['Peak Hours Indicator'] = np.where((df['Total day minutes'] >= 9*60) & (df['Total day minutes'] <= 17*60), 1, 0)
```

Dataset after Creating features

Daytime Minutes Percentage	Evening Minutes Percentage	Night Minutes Percentage	Total Calls	Total Charges	Intl Plan Interaction	Voicemail Plan Interaction	Peak Hours Indicator
37.485860	27.912896	34.601244	300	75.56	0.0	25	0
26.426819	31.970564	41.602617	329	59.24	0.0	26	0
46.168437	22.989378	30.842185	328	62.29	0.0	0	0

Step 2: Bucketing Numerical Features

Imagine we're dealing with numbers of customer service calls. Instead of dealing with each number individually, we group them into categories like 'low', 'medium', and 'high'.

```
# Bucketing Customer Service Calls
df['Customer Service Calls Bucketed'] = pd.cut(df['Customer service calls'],
                                               bins=[-1, 2, 5, np.inf],
                                               labels=['low', 'medium', 'high'])
print("\nDataset After Bucketing Customer Service Calls:")
```

Dataset after bucketing

```
Dataset After Bucketing Customer Service Calls:
```

Total day minutes	Total day calls	Total day charge	Total eve minutes	...	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn	Customer Service Calls Bucketed
265.1	110	45.07	197.4	...	16.78	244.7	91	11.01	10.0	3	2.70	1	FALSE	low
161.6	123	27.47	195.5	...	16.62	254.4	103	11.45	13.7	3	3.70	1	FALSE	low
243.4	114	41.38	121.2	...	10.30	162.6	104	7.32	12.2	5	3.29	0	FALSE	low
299.4	71	50.90	61.9	...	5.26	196.9	89	8.86	6.6	7	1.78	2	FALSE	low
166.7	113	28.34	148.3	...	12.61	186.9	121	8.41	10.1	3	2.73	3	FALSE	medium

Step 3: Transforming Categorical Variables:

Here, categorical variables like 'International plan' and 'Voice mail plan' are transformed using Label Encoding, converting 'Yes' to 1 and 'No' to 0. Additionally, 'Customer Service Calls Bucketed' is encoded for further analysis.

```
# Transforming categorical variables
label_encoder = LabelEncoder()
df['International plan'] = label_encoder.fit_transform(df['International plan'])
df['Voice mail plan'] = label_encoder.fit_transform(df['Voice mail plan'])
df['Customer Service Calls Binned'] = label_encoder.fit_transform(df['Customer Service Calls Bucketed'])
```

Dataset after transforming

Dataset After Transforming Categorical Variables:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes
0	KS	128	415	0	1	25	265.1	110	45.07	197.4
1	OH	107	415	0	1	26	161.6	123	27.47	195.5
2	NJ	137	415	0	0	0	243.4	114	41.38	121.2
3	OH	84	408	1	0	0	299.4	71	50.90	61.9
4	OK	75	415	1	0	0	166.7	113	28.34	148.3

Step 4: Feature Selection

We refined our dataset by focusing on key features that significantly influence customer churn prediction. This involved excluding unnecessary columns, encoding categorical variables, and leveraging statistical methods to pinpoint the top 10 impactful features.

```
X = df.drop(['Churn', 'Customer Service Calls Bucketed', 'State'], axis=1)
y = df['Churn']
X_encoded = pd.get_dummies(X, columns=['Customer Service Calls Binned'])
selector = SelectKBest(score_func=f_classif, k=10)
X_selected = selector.fit_transform(X_encoded, y)

# Get selected feature names
selected_features = X_encoded.columns[selector.get_support()]
```

Selected features

```
Selected Features:
Index(['Daytime Minutes Percentage', 'Evening Minutes Percentage',
       'Night Minutes Percentage', 'Total Calls', 'Total Charges',
       'Intl Plan Interaction', 'Voicemail Plan Interaction',
       'Peak Hours Indicator', 'Customer Service Calls Binned_0',
       'Customer Service Calls Binned_1'],
```