Feature Engineering: Discussing the importance of feature selection and extraction in improving model performance in python.

Feature engineering is a crucial aspect of building interpretable machine learning models. It involves crafting new features or modifying existing ones from raw data to enhance the model's interpretability and predictive performance.

In this article, we will delve into the world of Feature Engineering and the approach of it in data-driven world. Below is the table of content of this article:

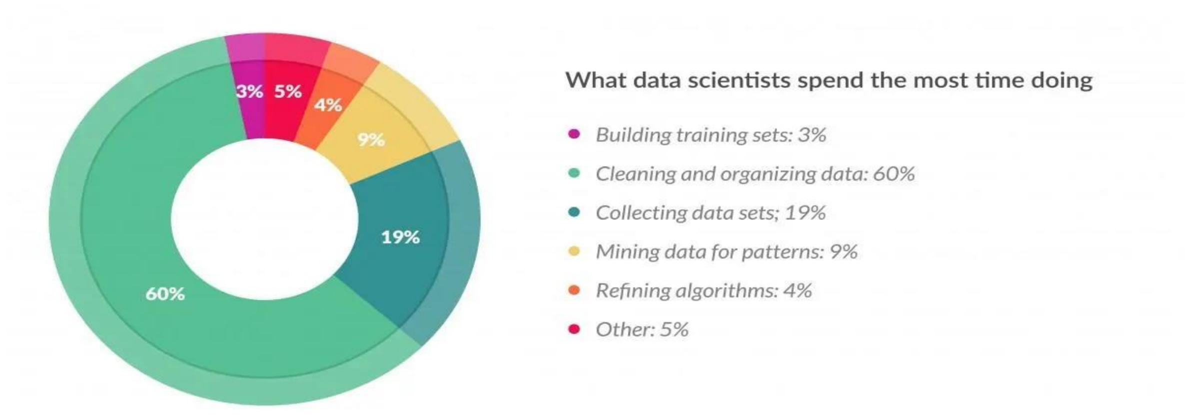
1) Definition:

Feature engineering is process of transforming raw data into a format that is easier to interpret for model training. It involves manipulating existing features or creating new ones to improve the performance of machine learning algorithms.

2) Importance:

1) **Noise reduction:** Eliminating extra variables helps in model performance as it removes redundant and unnecessary data and boosts the learning speed.
2) **Improve Performance:** Well-engineered features can significantly boost the predictive accuracy of machine learning models, leading to more reliable insights and decisions.
3) **Interpretability:** By transforming raw data into meaningful features, feature engineering can enhance the interpretability of machine learning models, facilitating a deeper understanding of the underlying relationships within the data.

3) More facts about feature engineering:



According to a survey in Forbes, data scientists spend **80% of their time** on data preparation.

Data scientists spend –

- 60% of the time in organizing and cleaning data.
- 19% of the time is spent in collecting datasets.
- 9% of the time is spent in mining the data to draw patterns.
- 3% of the time is spent in training the datasets.
- 4% of the time is spent in refining the algorithms.
- 5% of the time is spent in other tasks.

Feature engineering requires a lot of domain knowledge and proper implementation can help different domain in different ways. For example –

- Companies like Amazon, GE Healthcare, and PayPal leverage feature engineering to enhance customer experience, predict disease progression, and safeguard financial systems.
- In finance, feature engineering detects suspicious patterns, ensuring secure transactions, while in education, it personalizes learning experiences for students.
- Music streaming services like Spotify use feature engineering to curate personalized playlists, and automotive companies like Tesla engineer features for autonomous vehicles.

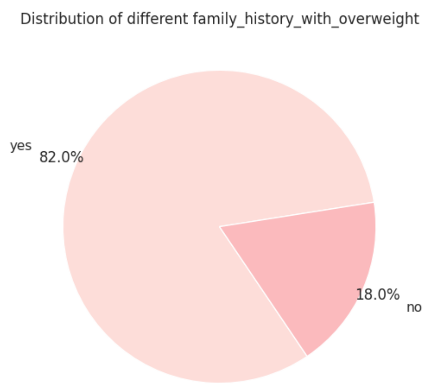
4) Pre-requisites:

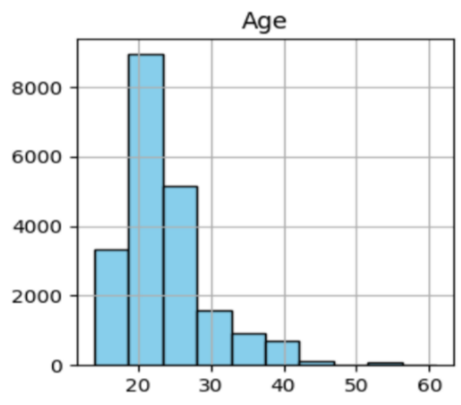
Data cleaning: This step includes basic data cleaning such as handling missing values, duplicates, outliers and data inconsistencies (wrong data format, wrong data type).

Basic EDA: Exploratory Data Analysis helps in understanding different variables present in the data. This step includes: -

1) Univariate analysis – Understanding one variable at a time.

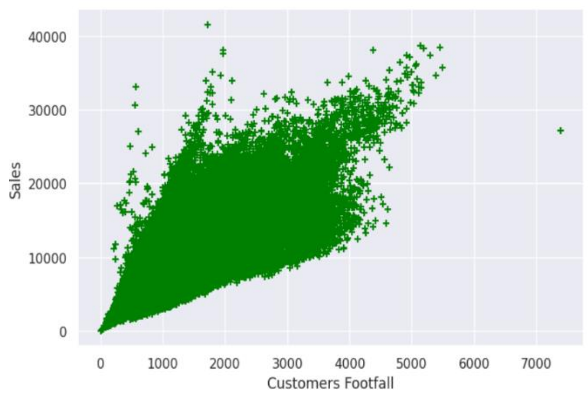
- If a variable is continuous, we could find out its distribution and potential outliers using histogram and box plot.
- If it is a categorical, we could use count plot or pie chart to find out whether it is balanced or not.

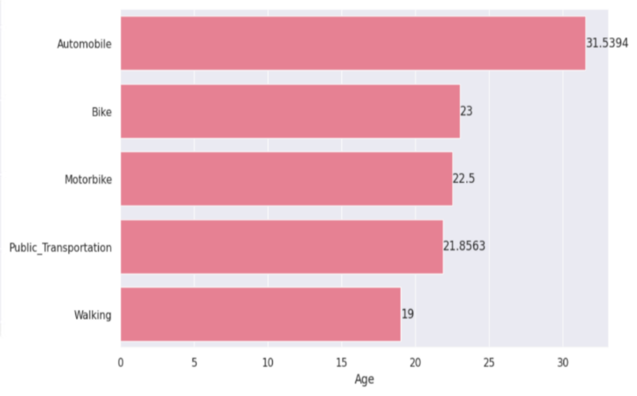




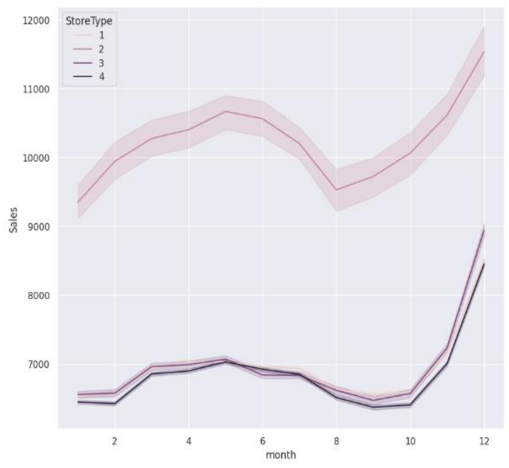
2) Bivariate analysis – Finding out relationship between two variables at a time.

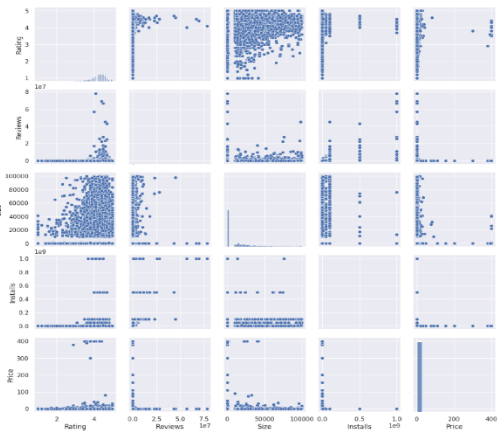
- If one variable is continuous and other is categorical – We could us group by method on categorial column to find out mean, median, sum etc. on each of the category of that column. Charts can be used – Bar chart, column chart, violin plot, box plot.
- If both the variables are continuous – We could use scatter plot. Other thing we could also do is to bin one column and then make a bar chart. But this method usually works if one variables' data points are concentrated to some absolute values.
- If both the variables are categorical – We could use bar chart in this condition by grouping by one of the columns and find out the count of another column.





3) Multivariate analysis - Finding out relationship between three or more variables at a time. We could use scatter plot, stacked bar chart, correlation heatmap, pair plot, kdeplot etc.





5) Methods of Feature Engineering:

Feature selection –

This involves identifying and retaining the most relevant features while discarding redundant or irrelevant ones. Some common methods of feature selection are –

- Univariate Feature Selection (SelectKBest Method)
- Recursive Feature Elimination
- Feature Importance from Trees
- Variance Inflation Factor (VIF)
- Using Correlation Heatmap

Feature Extraction –

Feature extraction involves creating new features from the existing ones through techniques such as:

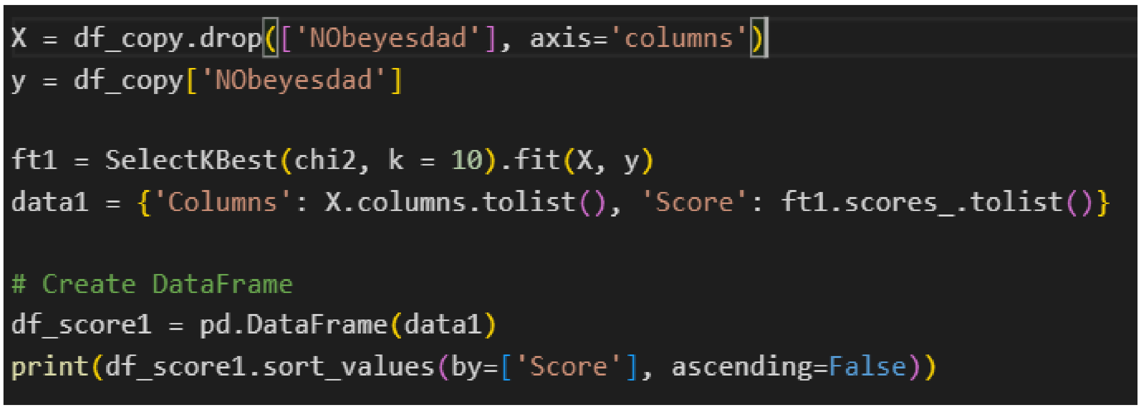
- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- t-distributed Stochastic Neighbour Embedding (t-SNE)

Feature Construction –

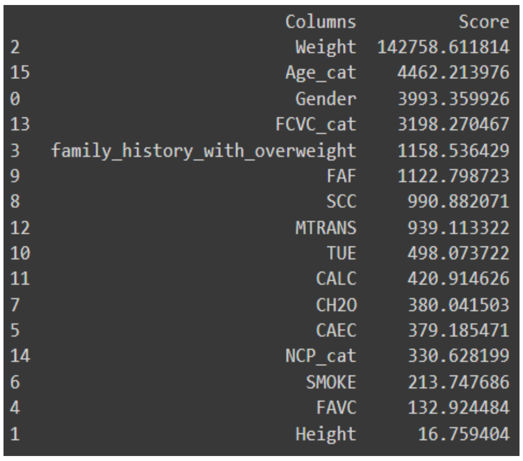
This entails creating new features by combining or transforming existing ones based on domain knowledge or heuristic approaches. We could also use binning to convert continuous feature to categorical.

Let’s understand each of them:

Univariate Feature Selection - It operates by evaluating each feature individually with respect to the target variable, without considering the interactions between features, to identify features that have the strongest relationship with the target variable. We can use SelectKBest Method: For continuous target variables (Regression problem), commonly used tests include the F-test or mutual information. For categorical target variables (Classification problem), the chi-square test is often employed.

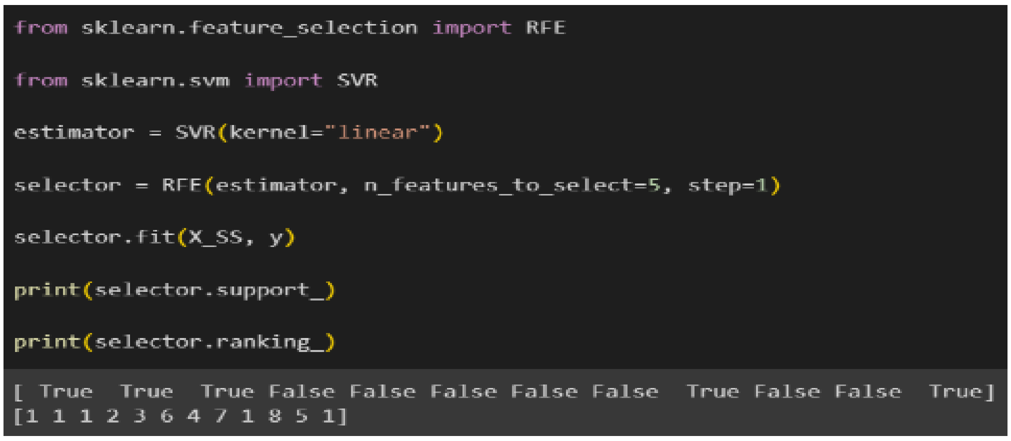


```
X = df_copy.drop(['NObeyesdad'], axis='columns')
y = df_copy['NObeyesdad']

ft1 = SelectKBest(chi2, k = 10).fit(X, y)
data1 = {'Columns': X.columns.tolist(), 'Score': ft1.scores_.tolist()}

# Create DataFrame
df_score1 = pd.DataFrame(data1)
print(df_score1.sort_values(by=['Score'], ascending=False))
```



```
                         Columns          Score
2                         Weight  142758.611814
15                       Age_cat    4462.213976
0                         Gender    3993.359926
13                      FCVC_cat    3198.270467
3   family_history_with_overweight    1158.536429
9                            FAF    1122.798723
8                            SCC     990.882071
12                        MTRANS     939.113322
10                           TUE     498.073722
11                          CALC     420.914626
7                           CH2O     380.041503
5                           CAEC     379.185471
14                       NCP_cat     330.628199
6                          SMOKE     213.747686
4                           FAVC     132.924484
1                         Height      16.759404
```

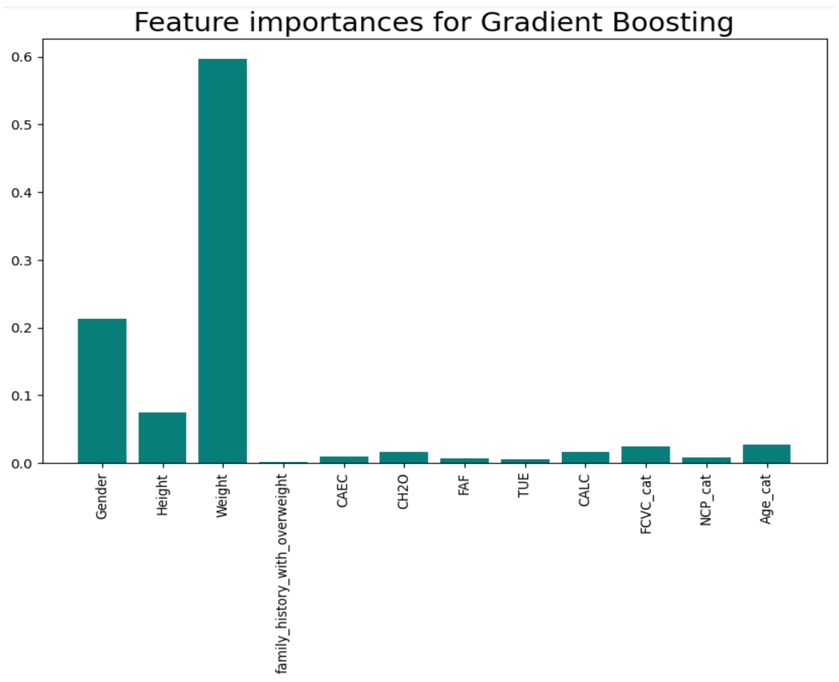
Recursive Feature Elimination - It operates by iteratively training the model on subsets of features, ranking them based on their importance, and eliminating the least significant features until the desired number of features is reached or a predefined performance criterion is met. (Suitable for both classification and regression problem).



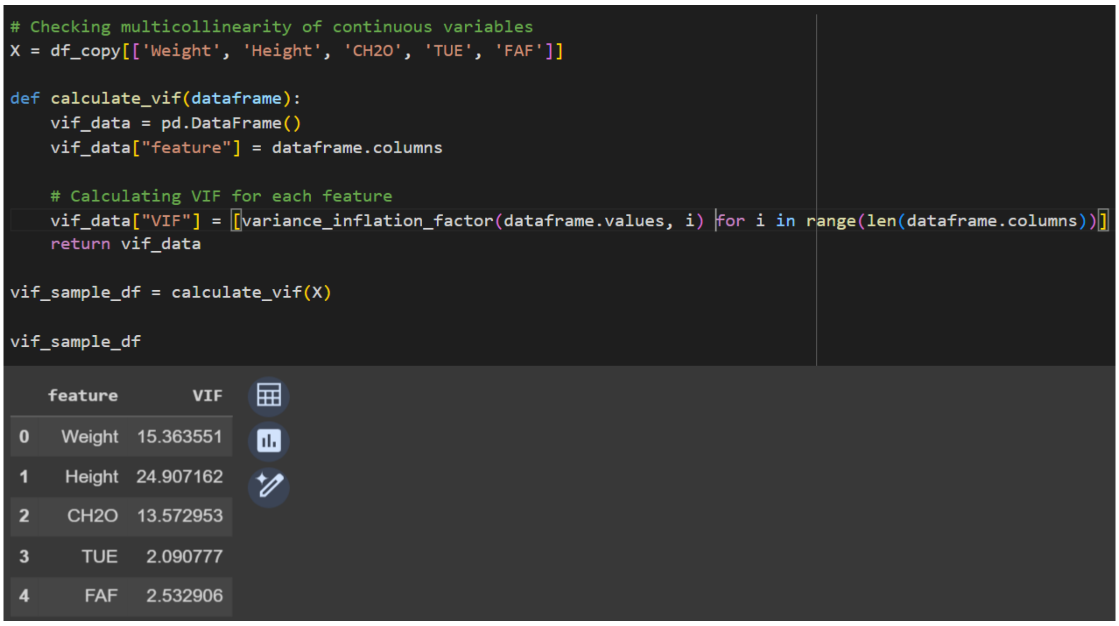
```
from sklearn.feature_selection import RFE

from sklearn.svm import SVR

estimator = SVR(kernel="linear")

selector = RFE(estimator, n_features_to_select=5, step=1)

selector.fit(X_SS, y)

print(selector.support_)

print(selector.ranking_)
```

```
[ True  True  True False False False False False  True False False  True]
[1 1 1 2 3 6 4 7 1 8 5 1]
```

Feature importance from trees – It refers to the process of assessing the contribution of individual features in decision tree-based models, such as Random Forests or Gradient Boosted Trees, towards predictive performance. This information can then be leveraged for feature selection. (Suitable for both classification and regression problem)



Variance Inflation Factor - VIF is a metric used to assess the severity of multicollinearity in regression analysis. Multicollinearity occurs when independent variables in a regression model are highly correlated with each other, which can lead to inaccurate and unreliable estimates of the regression coefficients. (Suitable for regression problem)

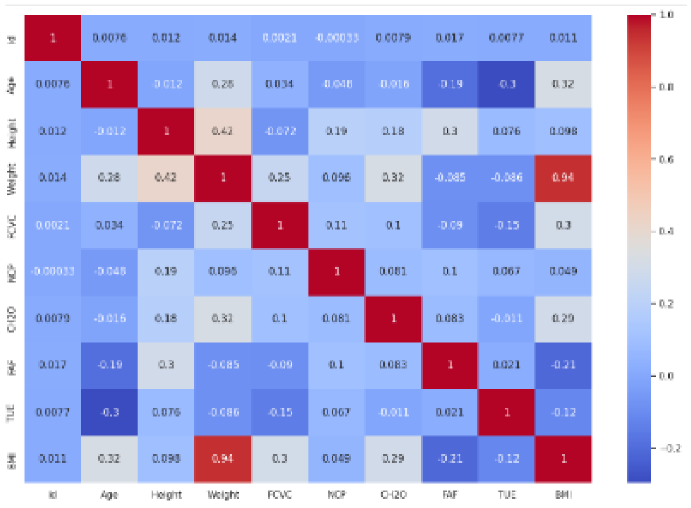


```
# Checking multicollinearity of continuous variables
X = df_copy[['Weight', 'Height', 'CH2O', 'TUE', 'FAF']]

def calculate_vif(dataframe):
    vif_data = pd.DataFrame()
    vif_data["feature"] = dataframe.columns

    # Calculating VIF for each feature
    vif_data["VIF"] = [variance_inflation_factor(dataframe.values, i) for i in range(len(dataframe.columns))]
    return vif_data

vif_sample_df = calculate_vif(X)

vif_sample_df
```

	feature	VIF
0	Weight	15.363551
1	Height	24.907162
2	CH2O	13.572953
3	TUE	2.090777
4	FAF	2.532906

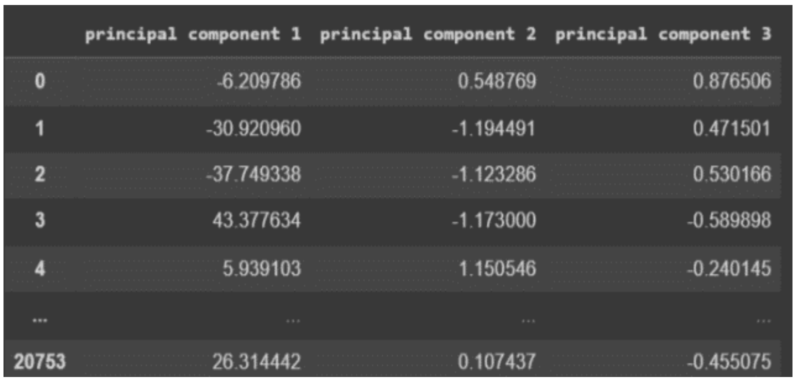
Using Correlation Heatmap - It uses colour gradients to depict the strength and direction of the correlation coefficients, making it easy to identify patterns of association between variables. It can used to find which variable depicts strong relation with target variable as well as independent variables (detecting multicollinearity). Note – Correlation heatmap can only show linear relationship but fails to detect complex relations that can happen between two or more variables. (Suitable for regression problem)

```
fig = plt.figure(figsize =(15, 10))
Correlation_Heatmap = sns.heatmap(df.corr(), cmap='coolwarm', annot=True)
```



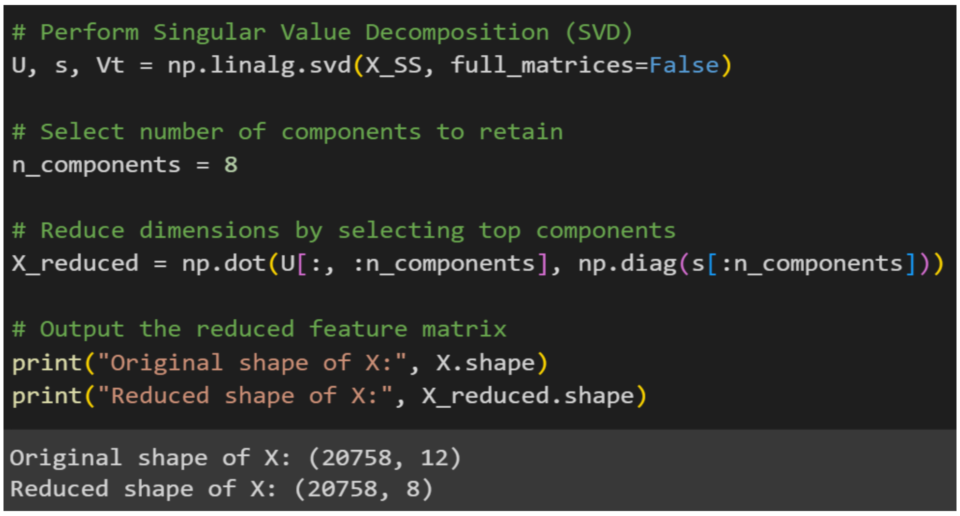
Principal Component Analysis (PCA) - is a dimensionality reduction technique commonly used for feature extraction in machine learning. It aims to transform a dataset consisting of possibly correlated variables into a set of linearly uncorrelated variables called principal components. By retaining a subset of the principal components that capture the most variance in the data, PCA effectively reduces the number of features while preserving as much information as possible.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=3)

principalComponents = pca.fit_transform(X)

principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2',
                          'principal component 3'])

principalDf
```



	principal component 1	principal component 2	principal component 3
0	-6.209786	0.548769	0.876506
1	-30.920960	-1.194491	0.471501
2	-37.749338	-1.123286	0.530166
3	43.377634	-1.173000	-0.589898
4	5.939103	1.150546	-0.240145
...	...	...	...
20753	26.314442	0.107437	-0.455075

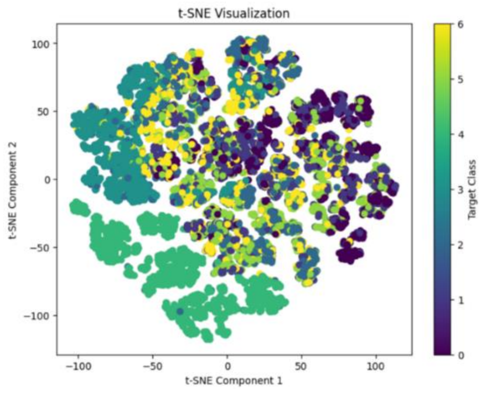
Singular Value Decomposition (SVD) – While there are many uses of SVD, it can also be used for dimensionality reduction and feature extraction by keeping only the most important singular values and associated singular vectors. By finding the basic patterns in the data and discarding the less important ones, SVD can help simplify the data and make it easier to work with.

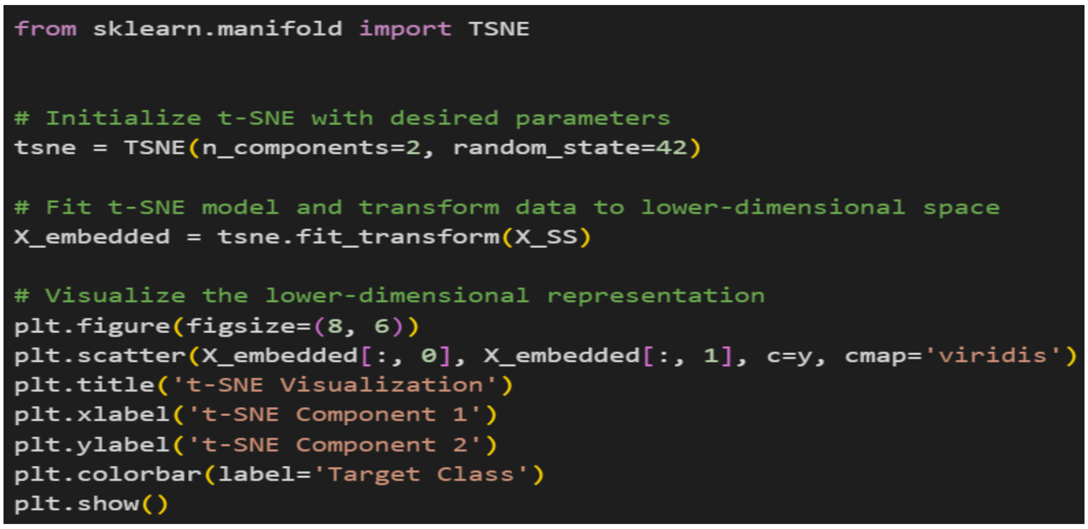


```
# Perform Singular Value Decomposition (SVD)
U, s, Vt = np.linalg.svd(X_SS, full_matrices=False)

# Select number of components to retain
n_components = 8

# Reduce dimensions by selecting top components
X_reduced = np.dot(U[:, :n_components], np.diag(s[:n_components]))

# Output the reduced feature matrix
print("Original shape of X:", X.shape)
print("Reduced shape of X:", X_reduced.shape)
```

```
Original shape of X: (20758, 12)
Reduced shape of X: (20758, 8)
```

t-distributed Stochastic Neighbour Embedding (t-SNE) – It is a nonlinear dimensionality reduction technique commonly used for data visualization and feature extraction. Unlike linear techniques such as PCA, t-SNE aims to preserve the local structure of the data in the lower-dimensional space.





```python
from sklearn.manifold import TSNE


# Initialize t-SNE with desired parameters
tsne = TSNE(n_components=2, random_state=42)

# Fit t-SNE model and transform data to lower-dimensional space
X_embedded = tsne.fit_transform(X_SS)

# Visualize the lower-dimensional representation
plt.figure(figsize=(8, 6))
plt.scatter(X_embedded[:, 0], X_embedded[:, 1], c=y, cmap='viridis')
plt.title('t-SNE Visualization')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.colorbar(label='Target Class')
plt.show()
```

6) Statistical Methods for Selecting Variables:

Continuous Data:

- Pearson Correlation Coefficient: Measures the linear correlation between two continuous variables.
- ANOVA (Analysis of Variance): Identifies significant differences in means among multiple groups.
- Lasso Regression: Employs L1 regularization to select relevant features while shrinking irrelevant ones to zero.

Categorical Data:

- Chi-Square Test: Determines the independence between categorical variables.
- Mutual Information: Measures the amount of information obtained about one variable through another.

Mixed Data Types:

- Mixed ANOVA: Handles datasets with both continuous and categorical variables by analysing interactions between them.
- ReliefF: An algorithm for feature selection that can handle both continuous and categorical attributes.

7) Conclusion:

Even though feature engineering is kind of overwhelming task, it is the one of the most crucial steps. By selecting, extracting, and constructing relevant features, data scientists can unlock the potential within complex datasets, leading to improved predictive performance and actionable insights. As we have seen there are many libraries and techniques available in python for feature engineering and selection for different supervised learning and data types. But a domain and subject matter knowledge plays an important too.

As organizations continue to harness the power of data-driven decision-making, mastering the art of feature engineering remains indispensable for unlocking the true potential of machine learning applications.