SILVER OAK UNIVERSITY

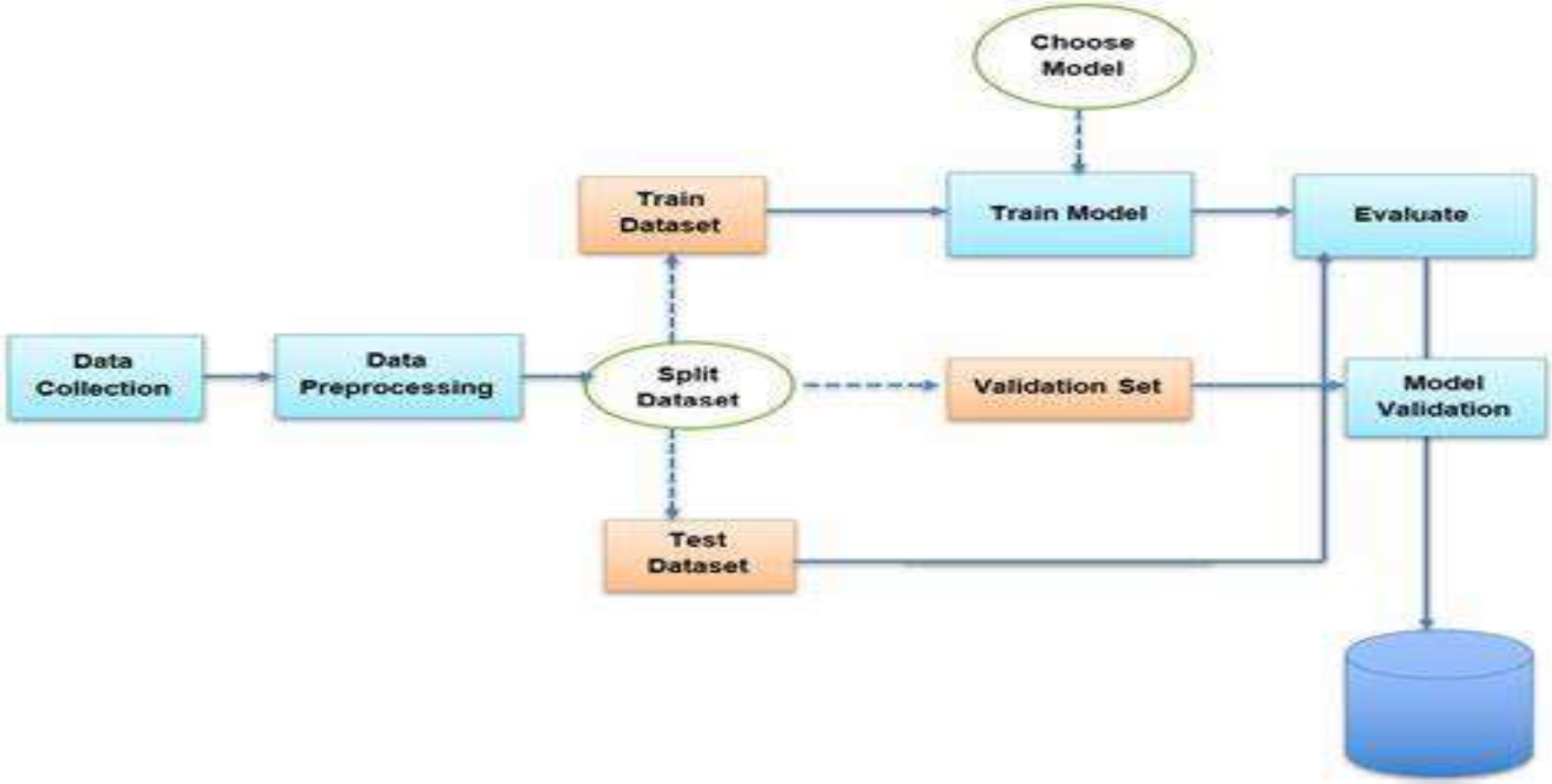
EDUCATION TO INNOVATION

SILVER OAK COLLEGE OF COMPUTER APPLICATION

SUBJECT :MACHINE LEARNING

TOPIC : Flow & Tools of machine learning

Machine Learning Workflow: From Data Collection to Model Deployment



- Machine learning is a powerful approach that enables computers to learn patterns from data and make predictions or decisions without explicit programming. The machine learning workflow consists of several key steps that guide the process of building and deploying effective machine learning models. Let's explore these steps in detail with examples:

- **Step 1: Data Collection**
  - The first step in the machine learning workflow is to gather relevant data that represents the problem we want to solve.
  - Example: Suppose we want to build a machine learning model to predict house prices. We collect data on various features of houses (e.g., size, number of bedrooms, location) and their corresponding prices.

- **Step 2: Data Preprocessing**
  - Once we have the data, we need to preprocess it to make it suitable for the machine learning algorithms.
  - Example: In the house price prediction example, we might need to handle missing values, scale numerical features, and encode categorical variables (e.g., convert location names into numerical values).
- **Step 3: Feature Engineering**
  - Feature engineering involves selecting or creating the most relevant features that contribute to the model's performance.
  - Example: In the house price prediction, we may derive additional features like the age of the house or the distance to the nearest school to improve the model's accuracy.

- **Step 4: Model Selection**
  - Model selection is about choosing the appropriate algorithm or model that fits the problem's requirements.
  - Example: For the house price prediction task, we could consider using regression models like linear regression or decision tree regression.

- **Step 5: Model Training**
  - Once we select the model, we need to train it on the preprocessed data to learn patterns and relationships.
  - Example: The selected model is trained using the historical data on house features and prices, so it learns how features influence house prices.

- **Step 6: Model Evaluation**
  - Model evaluation assesses the performance of the trained model on unseen data to ensure it generalizes well.
  - Example: We split the data into training and testing sets. The model's predictions on the test set are compared with the actual house prices to measure accuracy or other metrics.

- **Step 7: Hyper parameter Tuning**
  - Many models have hyper parameters (settings that control the learning process) that need to be fine-tuned for optimal performance.
  - Example: In a decision tree model, the maximum depth of the tree is a hyper parameter that affects the model's complexity. We adjust this parameter to improve accuracy.

- **Step 8: Model Deployment**
  - After selecting the best model, we deploy it in a real-world environment to make predictions on new data.
  - Example: The trained house price prediction model is deployed as a web application where users can enter house features, and the model predicts the price.

- **Step 9: Monitoring and Maintenance**
  - Once the model is deployed, it is essential to continuously monitor its performance and update it as needed.
  - Example: If the house price prediction model starts making inaccurate predictions due to changing market trends, we may need to retrain it with new data.

- **Conclusion:**

  The machine learning workflow involves several iterative steps, starting from data collection to deploying the trained model for real-world use. Each step is crucial for building accurate and reliable machine learning models. By following this workflow and continuously improving the model, we can leverage the power of machine learning to solve a wide range of problems effectively.

- Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data.
- A more general definition given by Arthur Samuel is – “Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.” They are typically used to solve various types of life problems.
- In the older days, people used to perform Machine Learning tasks by manually coding all the algorithms and mathematical and statistical formulas.
- This made the processing time-consuming, tedious, and inefficient. But in the modern days, it is become very much easy and more efficient compared to the olden days with various python libraries, frameworks, and modules.
- Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reasons is its vast collection of libraries.

- Python libraries that are used in Machine Learning are:

    - Numpy
    - Scikit-learn
    - Pandas
    - Matplotlib

- **NumPy** is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like Tensor Flow uses NumPy internally for manipulation of Tensors.
- NumPy is a Python library used for working with arrays.
- Functions:
    - array() : We can create a NumPy ndarray object by using the array() function.
    - concatenate() : we can join the array.
    - arange(): This function is used to create an array with a range of values.
    - array_split(): We use array_split() for splitting arrays, we pass it the array we want to split and the number of splits.
    - sort() : The NumPy ndarray object has a function called sort(), that will sort a specified array.
    - random.randint(): This function is used to create an array with random integer values between a specified range.
    - max(): This function is used to find the maximum value in an array.
    - min(): This function is used to find the minimum value in an array.
    - mean(): This function is used to find the mean value of an array.
    - median(): This function is used to find the median value of an array.
    - dot(): This function is used to find the dot product of two arrays.

- Example:

```
import numpy as np
# Creating two arrays of rank 2
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])

# Creating two arrays of rank 1
v = np.array([9, 10])
w = np.array([11, 12])

# Inner product of vectors
print(np.dot(v, w), "\n")

# Matrix and Vector product
print(np.dot(x, v), "\n")

# Matrix and matrix product
print(np.dot(x, y))
```

```
# Find Median
arr = np.array([1, 2, 3])
median_value = np.median(arr)
print(median_value)

# Find Mean
mean_value = np.mean(arr)
print(mean_value)

# Find Min
min_value = np.min(arr)
print(min_value)

# Find Max
max_value = np.max(arr)
print(max_value)
```

- Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML.
- Scikit-learn is a machine learning library for Python. It features several regression, classification and clustering algorithms including SVMs, gradient boosting, k-means, random forests and DBSCAN. It is designed to work with Python Numpy and SciPy.
- Functions:
  - **load_iris()** : if we want to load an Iris dataset
  - **head()** : this command returns the first five rows of the Iris Flower dataset.
  - **train_test_split()** : This method splits the data into train and test datasets.
  - **predict() :** Scikit-learn provides a wide range of machine learning algorithms that have a unified/consistent interface for fitting, predicting accuracy, etc.
  - **fit():** The classifier is trained using X_train data. The process is termed fitting. We pass the feature matrix and the corresponding response vector.

Example:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection  import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Create a Dataset
data = {
   'Hours_Studied': [1, 2, 3, 4, 5],
   'Scores': [10, 20, 30, 40, 50]
}
df = pd.DataFrame(data)
#  Split the dataset into training and testing sets.
X = df[['Hours_Studied']]
y = df['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and Train the Model
model = LinearRegression()
model.fit(X_train, y_train)

# Make Predictions
 y_pred = model.predict(X_test)

# Evaluate the Model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')

#Visualize the Results
plt.scatter(X, y, color='blue')
plt.plot(X, model.predict(X), color='red')
plt.xlabel('Hours Studied')
plt.ylabel('Scores')
plt.title('Hours Studied vs Scores')
plt.show()
```

- Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for grouping, combining and filtering data.
- It is one of the most used libraries in Python for data science or data analysis. It can read data from CSV or Excel files, manipulate the data, and generate insights from it. Pandas can also be used to clean data, filter data, and visualize data.
- Functions:
  - **read_csv() :** This function is used to retrieve data from CSV files in the form of a dataframe.
  - **Head():** This function is used to return the top n (5 by default) values of a data frame or series.
  - **Info():** This method is used to generate the summary of the DataFrame, this will include info about columns with their names, their datatypes, and missing values.
  - **Describe():** Returns descriptive statistics about the data like mean, minimum, maximum, standard deviation, etc.
  - **isNull() :** Returns the DataFrame/Series of the boolean values. Missing values gets mapped to True and non-missing value gets mapped to False.
  - **Sort_values() :** This method sorts the data frame in ascending or descending order of passed Column.
  - **Values_count() :** Returns the counts of the unique values in a series or from a dataframe’s column.
  - **Std() :** It returns sample standard deviation over the requested axis.

Example:

```
import pandas as pd

data = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],
        "capital": ["Brasilia", "Moscow", "New Delhi", "Beijing", "Pretoria"],
        "area": [8.516, 17.10, 3.286, 9.597, 1.221],
        "population": [200.4, 143.5, 1252, 1357, 52.98] }

data_table = pd.DataFrame(data)
print(data_table)
```

- Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
- Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc,
- Functions:
  - **plot() :** This function is used to create line charts, scatter plots, and other types of plots. It takes x and y values as inputs and can also accept optional arguments such as line style, marker style, and color. This function is the backbone of most Matplotlib visualizations.
  - **xlabel() & ylabel() :** These functions are used to label the x-axis and y-axis, respectively. They take a string as input and can also accept optional arguments such as font size, font weight, and color.
  - **title() :** This function is used to add a title to the plot. It takes a string as input and can also accept optional arguments such as font size, font weight, and color.
  - **grid() :** This function to add grid lines to the plot.
  - **scatter() :** This function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis.
  - **bar():** This function is used for Bar graph.

Example:

```
import matplotlib.pyplot as plt
# data to display on plots
x = [3, 1, 3]
y = [3, 2, 1]
plt.plot(x, y)
plt.title("Line Chart")
plt.legend(["Line"])
plt.show()
# This will plot a simple bar chart
plt.bar(x, y)
# Title to the plot
plt.title("Bar Chart")

# Adding the legends
plt.legend(["bar"])
plt.show()
```