AIMLOps-B4 Questions and Answers - Lab Session 6th October, Sunday

Question 1: For IPython.display do we need to install an additional library?

Answer 1: Here in google colab notebook it is already installed along with other libraries like numpy, pandas, sklearn. You don't need to install it additionally. If some libraries are not pre-installed then they can be installed using **pip**. For example, `!pip install numpy` **pip** is the Package Installer for Python. You can use pip to install packages from the PyPI(Python Package Index) and other indexes.

Question 2: For a Label encoder we do need to sort the Y column right, will it categorize the dataset by its own even if the values are mixed up in DF?

Answer 2: No, no need to sort the labels. Even if the values are mixed up LabelEncoder (or any other label encoder) will find the unique categories and assign numerical labels to it.

Question 3: Why are we doing `inversing the encoding` here? What is the difference between label_encoder.transform(Y) and then enc.inverse_transform(y)? What is the need of inverse transform?

Answer 3: ML models don't understand string texts. We need to convert the categorical features having text strings to numerical values before training/feeding to the model. This applies to the target column as well. We convert the class labels to numeric labels (for ex. Iris-setosa -> 0; Iris-verginica -> 1; Iris-versicolor-> 2), then train/feed to the model. Once the model is trained, we can use it to predict the numeric label for a new unknown sample(ex. new Iris flower). The output will be either 0,1, or 2. These numeric labels then again need to be mapped to class labels (Iris-setosa, Iris-veginica, Iris-versicolor). So, here the inversing of the encoding becomes important.
We don't have to remember these encoding mappings, the classes that do label encodings such as LabelEncoder provide methods to do that:

- *fit(...)*: to learn category mappings from the data passed
- *transform(...)*: to transform categories to numeric labels
- *fit_transform(...)*: to do fit and transform in one go
- *inverse_transform(...)*: to transform numeric labels back to category value

Question 4: What are the key methods that we need to understand for dataframe.

Answer 4: Here are some commonly used methods from the Pandas and NumPy libraries, which are essential for data analysis and manipulation in Python: Pandas Library (for DataFrames)

1. Creating a DataFrame:
   pd.DataFrame(data): Create a DataFrame from dictionaries, lists, arrays, or other data structures.
2. Basic DataFrame Methods:
   df.head(n): Displays the first n rows (default 5).
   df.tail(n): Displays the last n rows (default 5).
   df.shape: Returns the dimensions of the DataFrame (rows, columns).
   df.describe(): Provides descriptive statistics for numerical columns.
   df.info(): Displays concise information about the DataFrame.
   df.columns: Returns column names.
   df.index: Returns index of the DataFrame.
3. Selecting Data:
   df['column_name']: Selects a single column as a Series.
   df[['col1', 'col2']]: Selects multiple columns as a DataFrame.
   df.iloc[row_idx, col_idx]: Select data by row and column indices.
   df.loc[row_label, col_label]: Select data by row and column labels.

4. Filtering Data:
   df[df['column'] > value]: Filters rows based on condition.
   df.query('column > value'): Queries the DataFrame.

5. Missing Data:
   df.isnull(): Checks for missing values.
   df.dropna(): Drops rows with missing values.
   df.fillna(value): Fills missing values with a specified value.

6. Aggregation and Grouping:
   df.groupby('column').agg(func): Groups data by a column and applies aggregation function(s) (e.g., sum, mean).
   df.mean(), df.sum(), df.min(), df.max(): Aggregation methods for columns.

7. Sorting Data:
   df.sort_values('column'): Sorts DataFrame by values of a column.
   df.sort_index(): Sorts by the index.

8. Merging and Joining DataFrames:
   pd.merge(df1, df2, on='key'): Merges two DataFrames based on a key column.
   df1.join(df2): Joins DataFrames on their index.

These are the frequent methods of Pandas library that we need to use while working on any dataset. The choice of method will depend on the problem you are dealing with.

Question 5: Can we use vectorizer instead of label encoder ?

Answer 5: Not really.. like vectorizer will try to convert the text into embeddings ..wouldn't work here. We will see the Text Vectorizer in module5 NLP.

Question 6: What stratify = y does in the split method?

Answer 6: It ensures that the same distribution of class labels are maintained in train-set and test-set as in the entire dataset before splitting.

Question 7: Correct answer for the technical Question present in Assignment will be awarded a score of 3 points. Does this mean we'll be evaluated based on the technical questions given at the end only?

Answer 7: Yes. Evaluation will only be based on that technical question present in the assignment and not on the coding part. Code evaluation is part of mini-projects and will be done during the Mini-project sessions on Saturday Mornings.

Question 8: Can you please explain *random_state* and some examples of why it is used?

Answer 8: The `random_state` parameter is commonly used in machine learning libraries like scikit-learn to control the randomness of certain operations. It ensures that your results are reproducible, meaning that every time you run the code with the same `random_state`, you'll get the same output.
For example:
After splitting a dataset and training a model, you got 85% accuracy. You want to share this result with the other members/stakeholders, but if you don't know what was the random state of that split, then you are relying on luck that next time you will get the same result.

Question 9: Does stratify option used only for structured dataset, not for unstructured dataset?

Answer 9: The `stratify` option in functions like `train_test_split` is used primarily with labeled datasets, whether structured or unstructured. Its purpose is to ensure that the splits maintain the same proportion of classes as in the entire dataset.

Question 10: What are the key hyper-parameters used for Decision Tree Classifiers? What is their purpose?

Answer 10: The key hyper-parameters for DecisionTreeClassifier include:

1. **`max_depth`**: Limits the maximum depth of the tree to prevent overfitting. A shallower tree may generalize better.
2. **`min_samples_split`**: Specifies the minimum number of samples required to split an internal node. It controls the size of the leaf nodes.
3. **`min_samples_leaf`**: Sets the minimum number of samples that must be present in a leaf node. This helps smooth the model and prevent overfitting.
4. **`max_features`**: Determines the number of features to consider when looking for the best split. It can improve model generalization and reduce computation time.
5. **`criterion`**: Specifies the function to measure the quality of a split (e.g., `gini` or `entropy`). This influences how splits are determined.
6. **`max_leaf_nodes`**: Limits the number of leaf nodes in the tree, helping to simplify the model.

These hyper-parameters help control the complexity of the tree, impacting its performance and ability to generalize to unseen data.
You can check the hyperparameters of a particular model by visiting the model's sklearn documentation.

Question 11: How did the k-fold get selected by the GridSearchCV?

Answer 11: We can set the number of k consecutive folds we need by using the **cv** parameter of GridSearchCV class.

Question 12: What does the threshold mean in VarianceThreshold?

Answer 12: In VarianceThreshold, the threshold is a parameter that determines the minimum variance a feature must have to be retained. Features with variance below this threshold are considered low-variance and are removed from the dataset. This helps in feature selection by eliminating uninformative features.

For example, if a feature has the same value throughout all the samples(i.e. have variance=0) then this feature is redundant and will not be of any use while developing a model. We can set the threshold value greater than 0 such as 0.001 or 0.01, then it will remove these redundant features.