Department of Information Technology – University of the Punjab

Programming for AI – MPhil/PhD (AI) F22

Lab-09

Max Time: 2.5 hours Date: 01-03-2023

Instructions:

- Please provide your own solutions and DO NOT COPY the code from your colleagues or the web.
- You can discuss your problems only with the teachers.

Task # 01 - Pre Processing 12 x 5 = 60 Marks

1. Import all required libraries to implement the following functions, also create a 2D Numpy array that will be used in all these functions. **You need to use this same array in your program.**

   ```
   arr = np.array([[1, 2, np.nan], [3, np.nan, 5], [6, 7, 8]])
   ```

2. Write a function that takes a 2D Numpy array as input and imputes missing values using **SimpleImputer**. The function should take the following parameters as input: X (numpy array). **You will use this imputed array in the upcoming functions.**

3. Write a function that takes a 2D Numpy array and a tuple specifying the minimum and maximum range for the normalization as input. The function should normalize the array using **MinMaxScaler** and the specified range. Return the normalized Numpy array.

4. Write a function that takes a 2D Numpy array as input and transforms it using **QuantileTransformer** to normalize the data to a uniform distribution. The function should take the following parameters as input: ***X*** (numpy array), ***n_quantiles*** (integer), ***output_distribution*** (string), ***ignore_implicit_zeros*** (boolean), ***subsample*** (integer).

5. Write a function that takes a 2D Numpy array as input and transforms it using **PowerTransformer** to make it follow a normal distribution. The function should take the following parameters as input: ***X*** (numpy array), ***method*** (string), ***standardize*** (boolean), ***copy*** (boolean).

6. Write a function that takes a 2D Numpy array as input and normalizes it using **preprocessing.normalize**. The function should take the following parameters as input: ***X*** (numpy array), ***norm*** (string).

7. Write a function that takes a 2D Numpy array as input and performs ordinal encoding using **preprocessing.OrdinalEncoder**. The function should take the following parameters as input: ***X*** (numpy array). Your function should take care of the missing values.

8. Write a function that takes a 2D Numpy array as input and performs one-hot encoding using **preprocessing.OneHotEncoder**. The function should take the following parameters as input: ***X*** (numpy array).

9. Write a function that takes a 2D Numpy array as input and performs k-bins discretization using **preprocessing.KBinsDiscretizer**. The function should take the following parameters as input: ***X*** (numpy array), ***n_bins*** (integer), ***strategy*** (string).

10. Write a function that takes a 2D Numpy array as input and performs feature binarization using **preprocessing.Binarizer**. The function should take the following parameters as input: ***X*** (numpy array), ***threshold*** (float).

11. Write a function that takes a 2D Numpy array as input and creates polynomial features using **preprocessing.PolynomialFeatures**. The function should take the following parameters as input: ***X*** (numpy array), ***degree*** (int), ***include_bias*** (bool).

12. Write a function that takes a 2D Numpy array as input and creates spline features using **preprocessing.SplineTransformer**. The function should take the following parameters as input: ***X*** (numpy array), ***n_knots*** (int), ***degree*** (int).

After creating the required functions, paste the following in the next cell to test your functions.

```
# create a sample 2D numpy array

arr = np.array([[1, 2, np.nan], [3, np.nan, 5], [6, 7, 8]])

# replace the missing values with the most frequent value of the respective columns

arr_imputed = impute_data(arr)

# normalize the array using MinMaxScaler and the range (0, 1)

normalized_arr = normalize_array(arr_imputed, (0, 1))

# transform the array using QuantileTransformer

quantile_transformed_arr = quantile_transform_array(arr_imputed, n_quantiles=10,
output_distribution='uniform', ignore_implicit_zeros = False, subsample=100000)

# transform the array using PowerTransformer

power_transformed_arr = power_transform_array(arr_imputed, method = 'yeo-johnson',
standardize=False, copy=True)

# normalize the array using preprocessing.normalize

normalized_arr_preprocessing = normalize_array_preprocessing(arr_imputed, norm='l2')

# perform ordinal encoding using preprocessing.OrdinalEncoder

ordinal_encoded_arr = ordinal_encode_array(arr_imputed)
```

```
# perform one-hot encoding using preprocessing.OneHotEncoder

onehot_encoded_arr = onehot_encode_array(arr_imputed)

# perform k-bins discretization using preprocessing.KBinsDiscretizer

k_bins_discretized_arr = k_bins_discretize_array(arr_imputed, n_bins=3, strategy='uniform')

# perform feature binarization using preprocessing.Binarizer

binarized_arr = binarize_array(arr_imputed, threshold=0.5)

# create polynomial features using preprocessing.PolynomialFeatures

polynomial_features_arr = polynomial_features_array(arr_imputed, degree=2,
include_bias=True)

# create spline features using preprocessing.SplineTransformer

spline_transformed_arr = spline_transform_array(arr_imputed, n_knots=5, degree=3)

# print normalized_arr

print(arr_imputed)

# print normalized_arr

print(normalized_arr)

# print quantile_transformed_arr

print(quantile_transformed_arr)

# print power_transformed_arr

print(power_transformed_arr)

# print normalized_arr_preprocessing

print(normalized_arr_preprocessing)

# print ordinal_encoded_arr

print(ordinal_encoded_arr)

# print onehot_encoded_arr

print(onehot_encoded_arr)

# print k_bins_discretized_arr
```

```python
print(k_bins_discretized_arr)
# print binarized_arr
print(binarized_arr)
# print polynomial_features_arr
print(polynomial_features_arr)
# print spline_transformed_arr
print(spline_transformed_arr)
```