Big Data Technologies(CSP-554)
Final Project report

Tirumalesh Nagothi
04-19-2023

Title: Easing the data preprocessing by extending the pandas API in PySpark and integrating TensorFlow and boto3.

Contents

Development Summary

The development plan aims to contribute to Apache Spark by adding new features and techniques to PySpark. The focus is on adding encoding and imputing missing values using Pandas-API within Spark, integrating libraries such as boto3 and TensorFlow for cloud infrastructure and machine learning models, respectively.

To begin, we will clone the entire source code of Spark and relevant packages from GitHub and identify the appropriate path to add new features and integrate new packages. The next step involves testing the new features against simple datasets to assess the execution engine's response to the integration. This will help identify any issues that must be resolved before trying with larger datasets. Once the features are stable, the developer will run multiple test cases, including exception cases, to ensure the features are robust and improve their usage. To make it easy for others to understand, I've written inline documentation for each feature, including method and function descriptions.

This development plan demonstrates a structured and thorough approach to contributing to Apache Spark. The project covers testing, documentation, and integration with other libraries, ensuring the new features are reliable, efficient, and easily understood and adopted by other developers.

Objectives

The project objectives are to prepare comprehensive data for training the machine learning model by converting it into numerical form. To achieve this, the following tasks will be completed:

- Impute null values in the entire dataset using various methods to ensure that every data point is accounted for.
- Encode object-type values with numerical values using labelling with serial numbers, counting each unique value, and also acting like one-hot encoding (for Boolean features).
- Ensure that every value in the dataset is ready for training the machine learning model by performing data cleaning and pre-processing.
- Test the running of third-party libraries such as boto3 and TensorFlow's implementation to ensure their compatibility with the project's goals and objectives.

The project aims to produce clean, comprehensive data ready to train a machine learning model. The objectives prioritize data pre-processing and cleaning, followed by testing the compatibility of third-party libraries with the project's goals. This approach ensures that the final output is reliable, efficient, and easily used by other developers.

Next steps

In future, there will be many more techniques to come, and I will make this feature and integrations work dynamically so that every PySpark user can suitably access them. Since I started working with PySpark, I want to introduce more and more concepts that make the process of data pipelines much more accessible, which will save time.

Solution outline

The solution outline for the given problem can be summarized as follows:

1. Read the dataset into a PySpark Data Frame using the pandas API.
2. Define functions that the user can call to perform encoding operations. These functions should accept parameters such as the column's name to encode and any other relevant parameters.
3. Implement a function to encode the entire dataset if it contains object type columns.
4. Create an optional method to replace invalid values for each column or the entire dataset. This method should accept a simple string value from the user for string values imputation and the mode of imputation (mode, mean, or median) for numerical values.
5. Handle null values for numerical columns by imputing them using various methods such as mean, median, or mode.
6. Ensure the final output is a clean, comprehensive dataset ready for machine learning model training.
7. Test the implementation using sample data to validate that the encoding and imputation functions work as expected.

The solution outline proposes a comprehensive and efficient way to encode and pre-process data using PySpark's pandas API in Python. It ensures that the final output is reliable, efficient, and easily used by other developers.

Relevant literature

Feature engineering uses domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning and is both difficult and expensive. The need for manual feature engineering can be obviated by automated feature learning.

1. "Encoding Categorical Variables" by Max Halford (2018) - This article overviews various encoding techniques, including label encoding, and discusses their advantages and disadvantages.
3. "Categorical Encoding Using Label Encoding and One-Hot-Encoder" by GeeksforGeeks (2021) - This tutorial provides a step-by-step guide to implementing label encoding in Python and an explanation of how it works and when to use it.

5. "Label Encoding in Machine Learning" by Arvind N (2019) - This article explains label encoding, including its advantages and disadvantages. It provides examples of how it can be used in machine learning.

Distributed systems have become increasingly important in machine learning (ML) and deep learning (DL) as more extensive data sets and complex models require more computing power than a single machine can provide. TensorFlow is a popular framework for building and training ML and DL models, and it has robust support for distributed training. Here are some references for relevant literature for working on TensorFlow with distributed computing:

1. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems" by Martín Abadi et al. (2016) - This paper introduces TensorFlow distributed computing capabilities, including the use of parameter servers to coordinate model updates across multiple machines.
5. "Horovod: fast and easy distributed deep learning in TensorFlow" by Alex Sergeev et al. (2018) - This paper describes Horovod. This framework simplifies distributed training of deep learning models in TensorFlow by providing a simple API and efficient communication primitives.

These sources can provide valuable insights into working with TensorFlow and distributed systems, including best practices for scaling training jobs, setting up and managing a cluster, and optimizing performance. They can also help researchers to understand the benefits and challenges of distributed training and the trade-offs between different approaches.

Proposed system

The proposed system for achieving the objectives of adding new features and integrating new packages to PySpark can be broken down into several steps:

1. Clone the entire source code of Spark and other relevant packages from GitHub to a local development environment.
2. Identify the appropriate path within the source code where the new feature or package integration should be added.
3. Write the new feature or package integration using Python code compatible with PySpark.
4. Test the new feature or package integration against simple datasets to ensure the execution engine reacts as expected.
5. Test the new feature or package integration against larger datasets to identify any alternative modes of implementation that may be necessary for scalability.
6. Run multiple test cases, including exception cases, to improve the usage and identify potential bugs or errors.
7. Write inline documentation for each feature or function to make it easy to understand and use.
8. Submit the new feature or package integration as a pull request to the Spark GitHub repository for review and inclusion in future releases.

By following this proposed system, it is possible to add new features and integrate new packages into PySpark while ensuring that the execution engine remains stable and scalable. The testing and documentation steps help to ensure that the new functionality is practical and accessible to other developers while also minimizing the potential for errors or bugs.

Architecture

The system architecture would consist of multiple components that work together to achieve the desired functionality. The first component is the input module, which reads the dataset into a data frame. The user interacts with the system through the user interface module, where they can select the desired operations on the dataset. These operations include encoding the values of a particular column with the datatype as an object or string and encoding the entire dataset if it has any object-type columns.

The second component is the encoding module, which performs the actual encoding of the dataset based on the user's input. This module ensures that null values cannot be encoded while non-null values can be encoded. An optional method is to run the invalid value replacement for each column or the entire. This works by taking the simple string value of the user’s choice as a parameter for string values imputation and the mode of imputation, such as mode, mean or median for numerical values.

Finally, the output module is responsible for presenting the results to the user in a clear and understandable format. This module includes a reporting module that generates reports on the data encoding and imputation process, providing detailed information about the process, such as the encoding and imputation methods used, the size of the dataset, and the time taken to perform the operations.

Overall, the system architecture is designed to be scalable and flexible, allowing for adding new functionalities and methods as needed. It provides users an easy-to-use interface for data encoding and imputation, making it a valuable tool for data scientists and researchers.

Software components

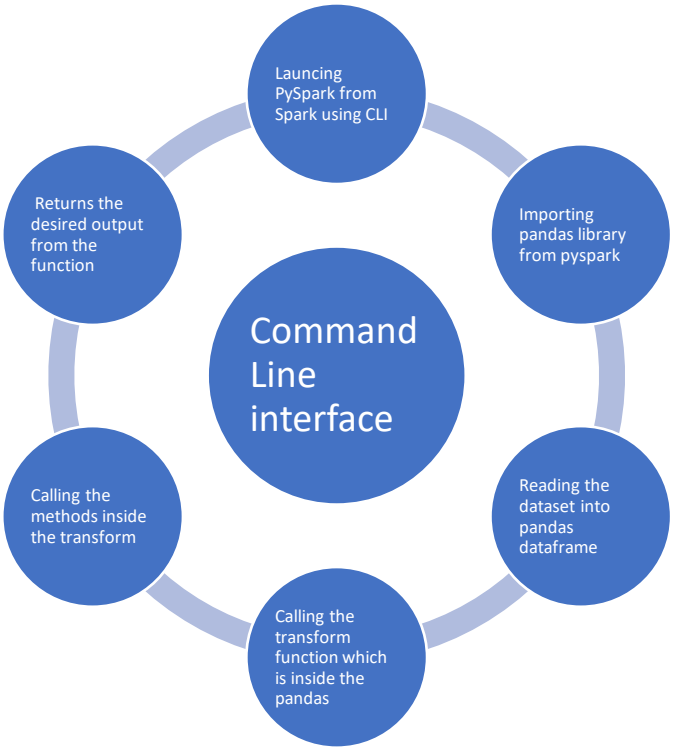
- Spark (PySpark)
- Pandas
- Python 3.9
- Command Shell
- GitHub
- Atom IDE

Interfaces

The PySpark interface runs on CLI(Command line interface) by accessing the spark folder in the site packages and running the command ./bin/pyspark.

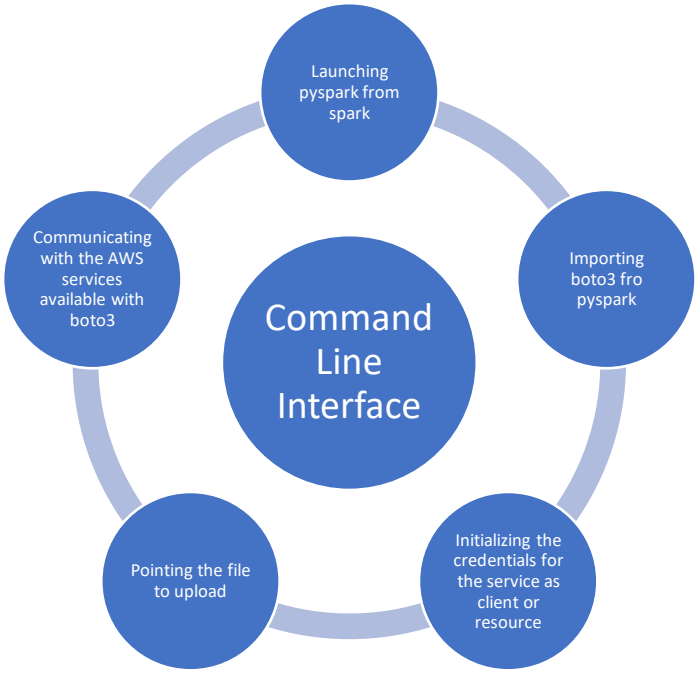
```
tirumaleshn2000@Tirumaleshs-MacBook-Pro spark % ./bin/pyspark
Python 3.9.6 (v3.9.6:db3ff76da1, Jun 28 2021, 11:49:53)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
23/04/19 20:16:41 WARN Utils: Your hostname, Tirumaleshs-MacBook-Pro.local resolves to a loopback address: 127.0.0.1; using 10.150.206.162 instead (on interface en0)
23/04/19 20:16:41 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/04/19 20:16:42 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.0-SNAPSHOT
      /_/

Using Python version 3.9.6 (v3.9.6:db3ff76da1, Jun 28 2021 11:49:53)
Spark context Web UI available at http://10.150.206.162:4040
Spark context available as 'sc' (master = local[*], app id = local-1681953402938).
SparkSession available as 'spark'.
2023-04-19 20:16:45.942669: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
>>>
```

Implementation



transform method

Running tensorflow on pyspark



Running boto3 on pyspark

Conclusion

Success/failure results

Transform test case 1:

```
from pyspark.pandas import transform
from pyspark import pandas as pd
import warnings
import os
warnings.filterwarnings('ignore')
from pydataset import data
datasets=data
```

```
def encoding(test_data):
    return transform.strd_to_numd(test_data)[0]

def null_replace(test_data):
    return transform.fill_null_data(test_data)
def mani(test_data):
    non_null_data=null_replace(test_data)
    return encoding(non_null_data)
try:
    os.mkdir('test_case_output_for_transform')
except:
    pass
cwd=os.getcwd()
def test_case(data,name):
    print('-------------------------------------------------------------------------------
-------------------------')
    print('-------------------------------------------------------------------------------
-------------------------')
    print('Dataset name:',name)
    print('----------------------------------------')
    test_data=data
    test1=encoding(test_data)

    del test1
    print('test 1 success')
    test2=null_replace(test_data)

    del test2
    print('test 2 success')
    test3=mani(test_data)

    del test3
    print('test 3 success')

for dataset in datasets()['dataset_id']:
    test_case(datasets(dataset),dataset)
```

Output:

```
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
Dataset name: Burt
----------------------------------
test 1 success
test 2 success
test 3 success
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
Dataset name: CanPop
----------------------------------
test 1 success
test 2 success
test 3 success
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
Dataset name: Chile
----------------------------------
------------------------------------------------------------
education  has null values.  Please replace the null values.
Do you want to the system to replace the null values for education and then encode the values[Y/n]?: y
------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
------------------------------------------------------------
vote  has null values.  Please replace the null values.
Do you want to the system to replace the null values for vote and then encode the values[Y/n]?: y
------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
test 1 success
Couldn't convert the values for the column education with datatype object. Please consider to convert the values to string format if you want to treat it as object dtype before running this function
Couldn't convert the values for the column vote with datatype object. Please consider to convert the values to string format if you want to treat it as object dtype before running this function
test 2 success
Couldn't convert the values for the column education with datatype object. Please consider to convert the values to string format if you want to treat it as object dtype before running this function
Couldn't convert the values for the column vote with datatype object. Please consider to convert the values to string format if you want to treat it as object dtype before running this function
------------------------------------------------------------
education  has null values.  Please replace the null values.
Do you want to the system to replace the null values for education and then encode the values[Y/n]?: n
----------------------------------------
vote  has null values.  Please replace the null values.
Do you want to the system to replace the null values for vote and then encode the values[Y/n]?:
Invalid Option
------------------------------------------------------------
vote  has null values.  Please replace the null values.
Do you want to the system to replace the null values for vote and then encode the values[Y/n]?: y
------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
test 3 success
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
Dataset name: Chirot
----------------------------------
test 1 success
test 2 success
test 3 success
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
Dataset name: Cowles
----------------------------------
test 1 success
test 2 success
test 3 success
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
Dataset name: Davis
----------------------------------
test 1 success
test 2 success
test 3 success
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
```

Transform test case 2:

```
from pyspark.pandas import transform
from pyspark import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from os import listdir

def testing_transform(path):
    def find_csv_filenames( path_to_dir, suffix=".csv" ):
        filenames = listdir(path_to_dir)
        return [ filename for filename in filenames if filename.endswith( suffix )
]

    files_list=find_csv_filenames(path)
    def encoding(test_data):
        return transform.strd_to_numd(test_data)[0]

    def null_replace(test_data):
        return transform.fill_null_data(test_data)

    def mani(test_data):
        non_null_data=null_replace(test_data)
        return encoding(non_null_data)

    def test_case(file):
        print('-------------------------------------------------------------------------
----------------------------')
        print('-------------------------------------------------------------------------
----------------------------')
        print('File name:',file)
        print('-------------------------------------')
        test_data=pd.read_csv(path+file)
        test1=encoding(test_data)

        del test1
        print('test 1 success')
        test2=null_replace(test_data)

        del test2
        print('test 2 success')
        test3=mani(test_data)

        del test3
        print('test 3 success')

    for i in range(len(files_list)):
        file=files_list[i]
        try:
            test_case(file)
        except:
            print("Couldn't test for the file: ",file)

path_input=input('Enter the path: ')
testing_transform(path_input)
```

Output:

```
--------------------------------------------------------------------------------------------------------------
File name: iowa-electricity.csv
-----------------------------------
test 1 success
test 2 success
test 3 success
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
File name: la-riots.csv
-----------------------------------
test 1 success
test 2 success
test 3 success
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
File name: th-airports.csv
-----------------------------------
------------------------------------------------------------
municipality  has null values.  Please replace the null values.
Do you want to the system to replace the null values for municipality and then encode the values[Y/n]?: y
------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
------------------------------------------------------------
gps_code  has null values.  Please replace the null values.
Do you want to the system to replace the null values for gps_code and then encode the values[Y/n]?: y
------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
------------------------------------------------------------
iata_code  has null values.  Please replace the null values.
Do you want to the system to replace the null values for iata_code and then encode the values[Y/n]?: y
------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
------------------------------------------------------------
local_code  has null values.  Please replace the null values.
Do you want to the system to replace the null values for local_code and then encode the values[Y/n]?: y
------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
------------------------------------------------------------
home_link  has null values.  Please replace the null values.
Do you want to the system to replace the null values for home_link and then encode the values[Y/n]?: y
------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
------------------------------------------------------------
wikipedia_link  has null values.  Please replace the null values.
Do you want to the system to replace the null values for wikipedia_link and then encode the values[Y/n]?: y
------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
------------------------------------------------------------
keywords  has null values.  Please replace the null values.
Do you want to the system to replace the null values for keywords and then encode the values[Y/n]?: y
------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
test 1 success
Couldn't convert the values for the column last_updated with datatype datetime64[ns]. Please consider to convert the values to string format if you want to treat it as object dtype before running this function
test 2 success
Couldn't convert the values for the column last_updated with datatype datetime64[ns]. Please consider to convert the values to string format if you want to treat it as object dtype before running this function
test 3 success
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
```

Output for fill_null_str:

```
>>> from pyspark import pandas as pd
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/spark/python/pyspark/pandas/__init__.p
his environment variable to '1' in both driver and executor sides if you use pyarrow>=2.0.0. pandas-on-Spark will se
  warnings.warn(
>>> df=pd.DataFrame({'col1':['a','b','c','hello',None],'col2':[1,2,3,4,5],'col3':['hey','1','2','3','4']})
>>> from pyspark.pandas import transform
>>> transform.fill_null_str(df,'col1')
0          a
1          b
2          c
3      hello
4    Missing
Name: col1, dtype: object
>>> transform.fill_null_str(df,'col1','Null value')
0            a
1            b
2            c
3        hello
4    Null value
Name: col1, dtype: object
>>> transform.fill_null_str(df,'col3')
0    hey
1      1
2      2
3      3
4      4
Name: col3, dtype: object
>>>
```

Output for fill_null_data:

```
>>> df=pd.DataFrame({'col1':['a','b','c','hello',None],'col2':[1,2,3,4,None],'col3':['hey','1','2','3','4']})
>>> df
    col1  col2 col3
0      a   1.0  hey
1      b   2.0    1
2      c   3.0    2
3  hello   4.0    3
4   None   NaN    4
>>> transform.fill_null_data(df)
      col1  col2 col3
0        a   1.0  hey
1        b   2.0    1
2        c   3.0    2
3    hello   4.0    3
4  Missing   1.0    4
>>> transform.fill_null_data(data_frame=df, fill_null_sc='Missing value', nc_impute_type='mean')
           col1  col2 col3
0             a   1.0  hey
1             b   2.0    1
2             c   3.0    2
3         hello   4.0    3
4  Missing value   2.5    4
>>>
```

Output for strc_to_numc:

```
>>> df=pd.DataFrame({'col1':['a','b','c','hello',None],'col2':[1,2,3,4,5],'col3':['hey','1','2','3','4']})
>>> transform.strc_to_numc(df,'col1')
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: y
--------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
(0    1
1    2
2    3
3    4
4    0
Name: col1, dtype: object, {'Missing': 0, 'a': 1, 'b': 2, 'c': 3, 'hello': 4})
>>> transform.strc_to_numc(df,'col1')
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: n
(0        a
1        b
2        c
3    hello
4     None
Name: col1, dtype: object, {None})
>>>
```

```
>>> from pyspark import pandas as pd
>>> df=pd.DataFrame({'col1':['a','b','c','hello',None],'col2':[1,2,3,4,5],'col3':['hey','1','2','3','4']})
>>> pd.transform.strc_to_numc(df,'col1')
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: y
--------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
(0    1
1    2
2    3
3    4
4    0
Name: col1, dtype: object, {'Missing': 0, 'a': 1, 'b': 2, 'c': 3, 'hello': 4})
>>> pd.transform.strc_to_numc(df,'col1')
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: y
--------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing): a
The value(a) already exists in the column: col1
Do you still want to replace with the given value [Y/n]?: y
(0    0
1    1
2    2
3    3
4    0
Name: col1, dtype: object, {'a': 0, 'b': 1, 'c': 2, 'hello': 3})
>>> pd.transform.strc_to_numc(df,'col1')
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: y
--------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing): Null value
(0    1
1    2
2    3
3    4
4    0
Name: col1, dtype: object, {'Null value': 0, 'a': 1, 'b': 2, 'c': 3, 'hello': 4})
>>> pd.transform.strc_to_numc(df,'col1')
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: n
(0        a
1        b
2        c
3    hello
4     None
Name: col1, dtype: object, {None})
>>>
```

Output for strd_to_numd:

```
>>> df=pd.DataFrame({'col1':['a','b','c','hello',None],'col2':[1,2,3,4,5],'col3':['hey','1','2','3','4']})
>>> transform.strd_to_numd(df)
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: y
--------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing): Null value
(  col1  col2 col3
0    1     1    4
1    2     2    0
2    3     3    1
3    4     4    2
4    0     5    3, {'col1': {'Null value': 0, 'a': 1, 'b': 2, 'c': 3, 'hello': 4}, 'col3': {'1': 0, '2': 1, '3': 2, '4': 3, 'hey': 4}})
>>> transform.strd_to_numd(df)
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: n
(    col1  col2 col3
0      a     1    4
1      b     2    0
2      c     3    1
3  hello     4    2
4   None     5    3, {'col1': {None}, 'col3': {'1': 0, '2': 1, '3': 2, '4': 3, 'hey': 4}})
>>> transform.strd_to_numd(df)
--------------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: y
--------------------------------------------------
Enter the new value or press ENTER to assign the default value(missing):
(  col1  col2 col3
0    1     1    4
1    2     2    0
2    3     3    1
3    4     4    2
4    0     5    3, {'col1': {'Missing': 0, 'a': 1, 'b': 2, 'c': 3, 'hello': 4}, 'col3': {'1': 0, '2': 1, '3': 2, '4': 3, 'hey': 4}})
>>>
```

```
>>> df=pd.DataFrame({'col1':['a','b','c','hello',None],'col2':[1,2,3,4,5],'col3':['hey','1','2','3','4']})
>>> transform.strd_to_numd(df)
--------------------------------------------------------------------
col1  has null values.  Please replace the null values.
Do you want to the system to replace the null values for col1 and then encode the values[Y/n]?: y
-----------------------------------------------
Enter the new value or press ENTER to assign the default value(missing): Null value
(  col1  col2 col3
0    1     1    4
1    2     2    0
2    3     3    1
3    4     4    2
4    0     5    3, {'col1': {'Null value': 0, 'a': 1, 'b': 2, 'c': 3, 'hello': 4}, 'col3': {'1': 0, '2': 1, '3': 2, '4': 3, 'hey': 4}})
>>>
```

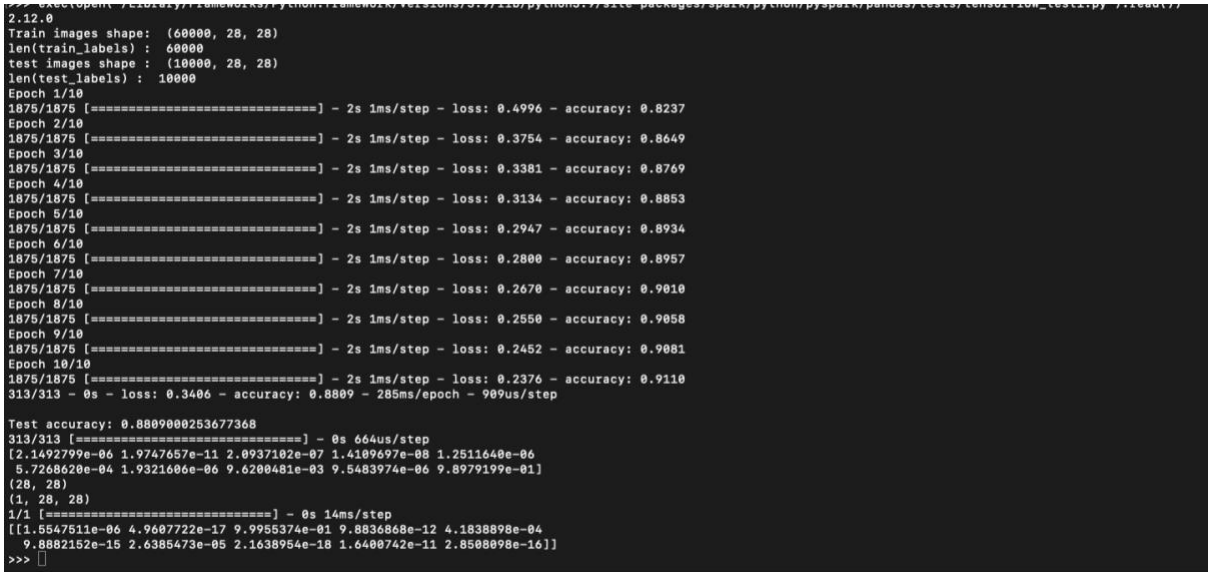
TensorFlow test case 1:

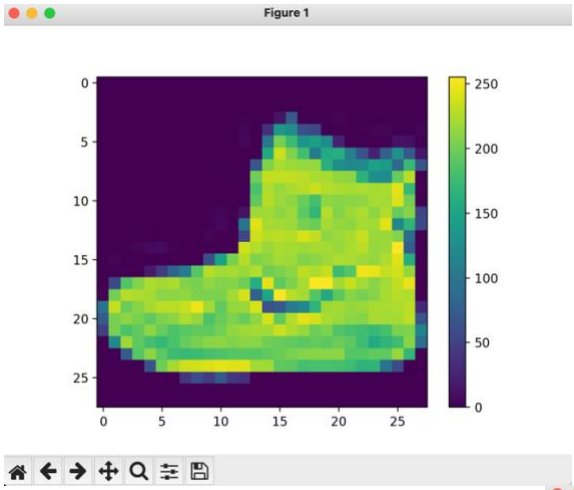
```
# TensorFlow and tf.keras
import pyspark.tensorflow as tf

import ssl

ssl._create_default_https_context = ssl._create_unverified_context

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)

fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

print('Train images shape: ',train_images.shape)
print('len(train_labels) : ',len(train_labels))
print('test images shape : ',test_images.shape)
print('len(test_labels) : ',len(test_labels))

plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()

train_images = train_images / 255.0

test_images = test_images / 255.0

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()

model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])



model.compile(optimizer='adam',
```

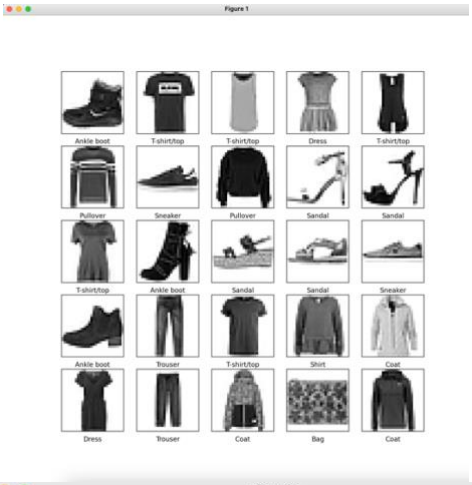
```
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])


model.fit(train_images, train_labels, epochs=10)

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)


probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])


predictions = probability_model.predict(test_images)

print(predictions[0])


np.argmax(predictions[0])


test_labels[0]

def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')


i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()


i = 12
plt.figure(figsize=(6,3))
```

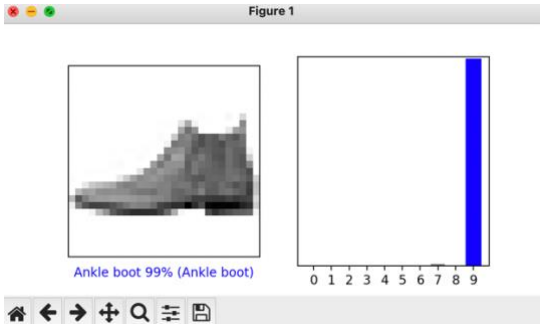
```
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()

# Grab an image from the test dataset.
img = test_images[1]

print(img.shape)

# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)


predictions_single = probability_model.predict(img)

print(predictions_single)



plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
plt.show()



np.argmax(predictions_single[0])
```

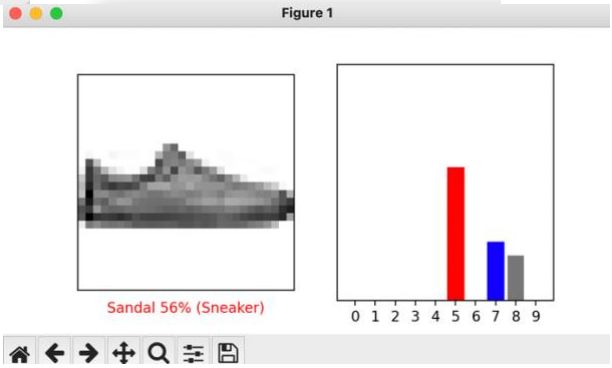
Output:

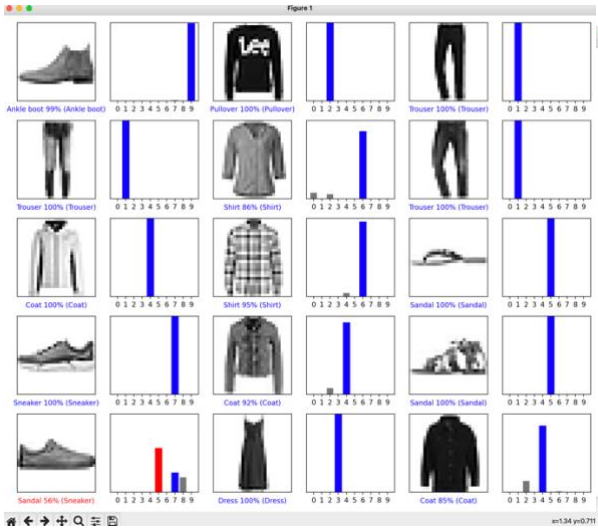


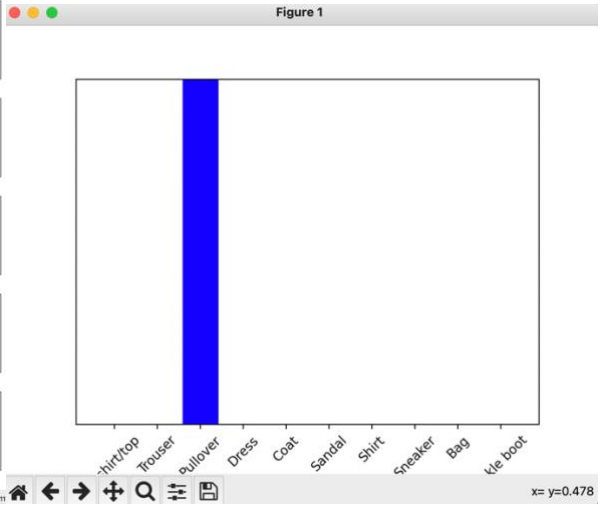










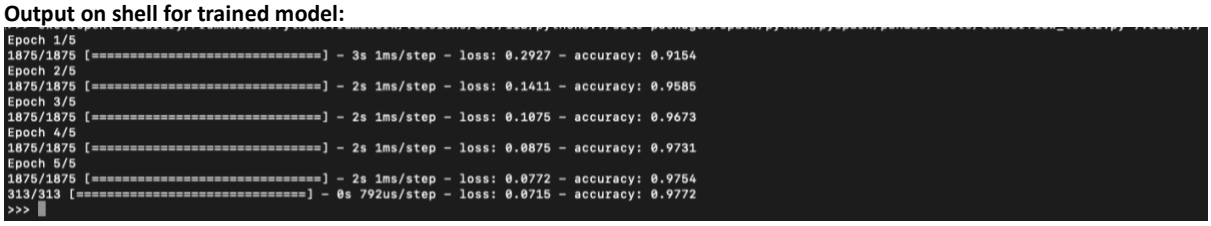


TensorFlow test case2:

```
import pyspark.tensorflow as tf
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
```

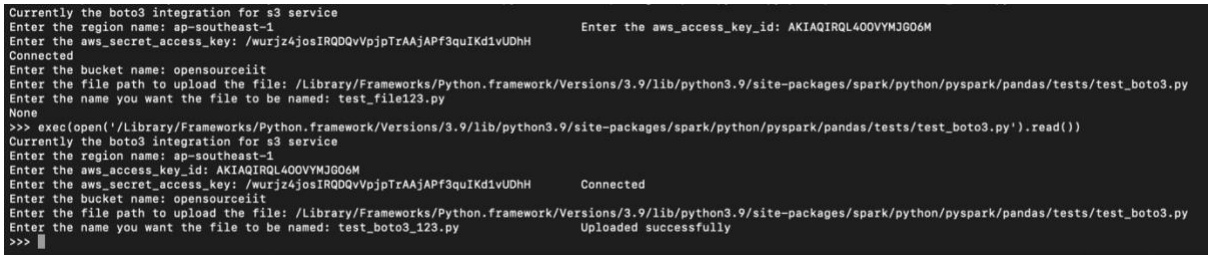
```
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Output on shell for trained model:

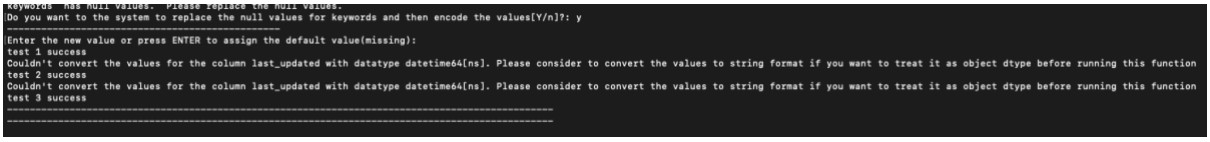


Test code for boto3:

```
from pyspark import boto3
print('Currently the boto3 integration for s3 service')
#taking the region anme, access keys from the user.
region_name=input('Enter the region name: ')
aws_access_key_id=input('Enter the aws_access_key_id: ')
aws_secret_access_key=input('Enter the aws_secret_access_key: ')
#intializing the connection as None
connection=None
try:
    #connecting the s3 service the given credentials to show the status as
connected if the conneciton is made.
    s3_client = boto3.resource(service_name='s3', region_name=region_name,
                                    aws_access_key_id=aws_access_key_id,
                                    aws_secret_access_key=aws_secret_access_key)
    print('Connected')
    connection='Success'
#in exception case, it will produce the error to show what is wrong in the given
credentials
except:
    s3_client = boto3.resource(service_name='s3', region_name=region_name,
aws_access_key_id=aws_access_key_id,
aws_secret_access_key=aws_secret_access_key)
#checking whether connection equal to 'Success'
if connection=='Success':
    #taking the bucket name, file path to upload and naming the file.
    bucket_name=input('Enter the bucket name: ')
    file_path=input('Enter the file path to upload the file: ')
    name_file=input('Enter the name you want the file to be named: ')
    result=s3_client.Bucket(bucket_name).upload_file(file_path,name_file)
    #if the result it None then the file got uploaded successfully
    if result==None:
        print('Uploaded successfully')
    #else there will be error produced.
```



Caveats/cautions



When executing the method to encode the values in the data frame. An exception will be raised if the column's type is other than the numeric or object. So, in that case, we recommend the user to convert that column to a string and apply the method again.

Also, when the boto3 is installed and copied inside the PySpark, there might be some errors that the s3transfer module needs to be found. In that case, it is recommended to delete all the boto3 files across the machine or uninstall them. So, when launching the PySpark engine, it automatically checks for the boto3 module inside the pyspark and proceeds with whether it needs to be installed.

Bibliography

My github repository master branch: **https://github.com/tirumaleshn2458/spark/tree/master/python/pyspark**
My github repository the latest branch: https://github.com/tirumaleshn2458/spark/tree/master_clone9/python/pyspark
TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems: **https://arxiv.org/abs/1603.04467**
Horovod: fast and easy distributed deep learning in TensorFlow: https://arxiv.org/abs/1802.05799
Target encoding done the right way: https://maxhalford.github.io/blog/target-encoding/
Feature encoding techniques – Machine learning: https://www.geeksforgeeks.org/feature-encoding-techniques-machine-learning/
Guide to Encoding Categorical Values in Python: https://www.niit.com/india/knowledge-centre/python-categorical-values
List of trending open-source big data projects: https://www.projectpro.io/article/best-open-source- big-data-projects-github/516
Getting started with spark: https://spark.apache.org
Pandas API on spark: https://spark.apache.org/docs/latest/api/python/getting_started/quickstart_ps.html
Spark GitHub repository: https://github.com/apache/spark.git
Boto3 GitHub repository: https://github.com/boto/boto3
TensorFlow GitHub repository: https://github.com/tensorflow/tensorflow
Building spark and launch pyspark from the source code on the local machine:
https://spark.apache.org/docs/latest/building-spark.html#building-submodules-individually
Guide to contributing to Open-Source projects: https://dev.to/codesphere/how-to-start-contributing-to-open-source-projects-on-github-534n
Integration with Cloud Infrastructures: https://spark.apache.org/docs/latest/cloud-integration.html

Code & data

Reference documentation

Software commands

To launch PySpark

1. cd /<spark directory path>
2. ./bin/pyspark

Inputs

1. For pd.transform.strc_to_numc(data,col_name), inputs are data_frame which is the pandas data frame, col_name which is the column name in the data frame.
2. For pd.transform.strd_to_numd(data_frame), inputs are data_frame.

Installation

Installation of PySpark is done in two ways:

- By cloning the source code from the spark GitHub repository and building using the "./build/mvn -Pmesos -DskipTests clean package"
- By installing using pip command, i.e. pip install pyspark

Installation of TensorFlow and boto3 and copying into PySpark:

```
import os
def run():
#installation for boto3
    try:
        import pyspark.boto3
    except:
        print('Installing boto3...')
        os.system('pip install boto3 -q -q -q')
        print('Installed boto3')
    import boto3
    boto3_path = os.path.dirname(boto3.__file__)
    import pyspark
    source_path = os.path.dirname(pyspark.__file__)
    boto3_folder = "boto3"
    try:
        boto3_source_path=os.path.join(source_path,boto3_folder)
    except:
        pass
    import shutil
    try:
        shutil.copytree(boto3_path,boto3_source_path)
    except:
        pass

#installation for tensorflow
    #import os
    try:
        import pyspark.tensorflow
    except:
        print('Installing tensorflow...')
        os.system('pip install tensorflow -q -q -q')
        print('Installed tensorflow')
    import tensorflow
    tensorflow_path = os.path.dirname(tensorflow.__file__)
    import pyspark
    source_path = os.path.dirname(pyspark.__file__)
    ts_folder = "tensorflow"
    try:
        tensorflow_source_path=os.path.join(source_path,ts_folder)
    except:
        pass

    import shutil
    #copying files in the tensorflow directory to the pyspark's tensorflow
directory
    try:
        shutil.copytree(tensorflow_path,tensorflow_source_path)
    except:

        pass
```

Testing framework

Cases

For new feature testing: Two test cases are done on the in-built datasets and different .csv files downloaded from the internet. Two scripts have been written for two other test cases.

For TensorFlow testing: Two test cases are done on the scripts from the official Tensorflow website.

Source code

Dependencies(open-source)

- Pandas, Numpy

- TensorFlow
- Boto3

Code for the introduced feature - transform.py :

Code for transform.fill_null_str method:

```
def fill_null_str(data_frame,col_name,replace_str_with='Missing'):
    col_data=data_frame[col_name].copy()
    while True:
        if replace_str_with in np.unique(col_data.dropna().unique().to_numpy()):
            print('The value({}) already exists in the column:
{}'.format(replace_str_with,col_name))
            ow_or_not=input('Do you still want to replace with the given value
[Y/n]?: ')
            if ow_or_not.lower()=='y':
                replaced_col=col_data.fillna(replace_str_with)
                return replaced_col
                break
            elif ow_or_not.lower()=='n':
                new_value=input('Enter the non-existing value to replace: ')
                if new_value in np.unique(col_data.dropna().unique().to_numpy()):
                    pass
                else:
                    replaced_col=col_data.fillna(replace_str_with)
                    return replaced_col
                    break
            else:
                print('Invalid input')
        else:
            replaced_col=col_data.fillna(replace_str_with)
            return replaced_col
```

Code for transform.fill_null_num method:

```
def fill_null_num(data_frame,col_name,impute_type='mode'):
    if impute_type=='mode':
        if len(data_frame[col_name].value_counts())==0:
            replaced_col=data_frame[col_name].fillna(0)
            return replaced_col
        else:

replaced_col=data_frame[col_name].fillna(data_frame[col_name].mode()[0].astype(floa
t))
            return replaced_col
    elif impute_type=='median':
        replaced_col=data_frame[col_name].fillna(data_frame[col_name].median())
        return replaced_col
    elif impute_type=='mean':
        replaced_col=data_frame[col_name].fillna(data_frame[col_name].mean())
        return replaced_col
    else:
        print('Impute type is not valid')
```

Code for transform.fill_null_data method:

```
def fill_null_data(data_frame,fill_null_sc='Missing',nc_impute_type='mode'):
    data=data_frame.copy()
    for col_name in data.columns:
        if data_frame[col_name].isnull().sum()>0:
            try:
                if data[col_name].dtype=='O':
                    data[col_name]=fill_null_str(data,col_name,fill_null_sc)
                else:
                    data[col_name]=fill_null_num(data,col_name,nc_impute_type)
            except:
                print("Couldn't convert the values for the column {} with datatype
{}. Please consider to convert the values to string format if you want to treat it
as object dtype before running this
function".format(col_name,data[col_name].dtype))
```

```
        else:
            pass
    return data
```

Code for transform.strc_to_numc method:

```
def strc_to_numc(data,col_name):
    col_data=data[col_name].copy()
    label_encode={None}
    if col_data.isnull().sum()>0:
        while True:
            print('-----------------------------------------------------------------
-------')
            print(col_name,' has null values. ','Please replace the null values.')
            replace_or_not=input('Do you want to the system to replace the null
values for {} and then encode the values[Y/n]?: '.format(col_name))
            if replace_or_not.lower()=='y':
                print('-----------------------------------------------')
                new_str=input('Enter the new value or press ENTER to assign the
default value(missing): ')
                if len(new_str)==0:
                    new_str='Missing'
                col_data=fill_null_str(data,col_name,replace_str_with=new_str)
                keys=np.unique(col_data.sort_values().to_numpy())
                values=range(len(keys))
                label_encode=dict(zip(keys,values))
                encoded_values=col_data.map(label_encode)
                return encoded_values,label_encode
                break
            elif replace_or_not.lower()=='n':
                return col_data,label_encode
                break
            else:
                print('Invalid Option')
                pass
    else:
        keys=np.unique(col_data.sort_values().to_numpy())
        values=range(len(keys))
        label_encode=dict(zip(keys,values))
        encoded_values=col_data.map(label_encode)
        return encoded_values,label_encode
```

Code for transform.strd_to_numd method:

```
def strd_to_numd(data_frame):
    dictionary_values={}
    data=data_frame.copy()
    for col_name in data.columns:
        if data[col_name].dtype=='O':
            try:
                converted_values=strc_to_numc(data,col_name)
                data[col_name]=converted_values[0]
                dictionary_values[col_name]=converted_values[1]
            except:
                data[col_name]=data[col_name].astype(str)
                converted_values=strc_to_numc(data,col_name)
                data[col_name]=converted_values[0]
                dictionary_values[col_name]=converted_values[1]
        else:
            pass
    return data,dictionary_values
```

Documentation

fill_null_str

fill_null_str is used to easily replace the null values for a string(object) column in the data frame. It can be considered the single line code for filling the null values with the user's string choice, making it more interactive.

Parameters

data_frame: Data frame object, Pandas data frame object, which is 2-dimensional.
col_name: String, Column name of the data_frame to get the null values filled.
replace_str_with: String, Default 'Missing'.
The value which should get replaced in place of the null value.

Returns

After replacing the null values, the series contains all the values in the column.

fill_null_num

fill_null_num replaces the null values in the numerical column of the data frame.
Filling the null values using a single line of code with the choice of imputation type.

Parameters

data_frame: Data frame object, Pandas data frame object, which is 2-dimensional.
col_name : String, Numerical column name of the data_frame to fill the null values.

impute_type: String, Default 'mode'
Type of imputation, i.e. whether the null values in the column should be replaced with the mode or mean or median value of the column.

Returns

Series
which contains all the values of the column after replacing the null values.

fill_null_data

fill_null_data is used to replace the null values for both the object and numerical columns in the data frame.
This is based on the fill_null_str and fill_null_num methods.

Parameters

data_frame: Data frame object, pandas data frame object which is 2-Dimensional.
fill_null_sc: String value, default 'Missing'
It takes the value(any) to replace with the null values in string columns of the data_frame.
nc_impute_type: String value, default 'mode.'
Takes the value(mode, mean, median) to apply the kind of imputation to replace the null values in numerical columns of the data_frame.

Returns

A pandas data frame object with no null values in the object and numerical columns.

strc_to_numc

strc_to_numc method encodes (labels) the string values in a column to the numerical values, i.e. Converts the entire object column's dtype to the numerical column.
This is also partially reliable on fill_null_str to replace the null values in the column since it cannot encode the columns with the null values.

Parameters

data: DataFrame object, which is the 2-dimensional pandas data frame object.
col_name: String value
Specific column's name you wish to encode the values in.

Returns

A Series object

Contains the encoded(numerical) values.

A dictionary object

Contains the key-value pairs that reference the numerical value(value) for each the string value(key) in the column.

strd_to_numd

strd_to_numd is used to encode all the string values in the columns to numerical values by applying labels for string values in columns.
This relies on the strc_to_numc method to encode a single column value in the data frame.

Parameters

data_frame: DataFrame object, which is the 2-dimensional pandas data frame object.

Returns

A data frame object

contains all the initial columns where the string(object) columns are encoded to numerical values.

A dictionary object

contains the dictionaries of each column's values and their labels returned from strc_to_numc.