data_cleaner_with_test_df_functionality.DataCleaner

class data_cleaner_with_test_df_functionality.DataCleaner(dataframe, features, test_df=None, mode="wnv")

Cleans the data provided in GA DSIF Project 4.

This class takes in a dataframe and a chosen feature list created from merged csv files from the GA DSIF project 4 assets, and creates an object with the attributes df, features, X, y, X_train, y_train, X_test, y_test.

If the use of a separate test df is intended, test df may be passed as an optional keyword argument.

Most attributes are initialised as None, and are filled by executing the clean() method on the object. The clean method contains a series of functions which start by removing outliers and populating the X and y attributes of the object based on the list of features passed in.

Each function thereafter is programmed to handle exceptions where the column has been removed due to it not being in X as a result of the features passed in. Please refer to the comments relating to each function below for a further explanation of what each function does.

The clean_test method may only be executed after the clean method has been executed as the encoder and scalers must first be fitted on training data.

The class object can be instantiated in two modes. In “num_mosquitos” mode, it is intended to clean (1) the train dataset with “NumMosquitos” as the target variable, and (2) the test dataset with the intention of predicting “NumMosquitos”. In “wnv” mode, it is intended to clean (1) the train dataset with “wnvpresent” as the target variable, and (2) the test dataset with the intention of predicting “wnvpresent”. This mode function was developed due to “NumMosquitos” being absent from the provided test dataset. The DataCleaner object will default to “wnv” mode if no argument is passed.

Parameters:	**dataframe: training dataframe** Training dataset in the form of a pandas dataframe. **features: list of features** Features to be processed in the form of a python list.

	test_df: testing dataframe, optional Testing dataset in the form of a pandas dataframe. **mode: {'wnv', 'num_mosquitos'}, default = 'wnv'** Sets the mode which the class constructor will operate in.
Attributes:	**df: dataframe** Training dataframe passed at instantiation. **test_df: test_df** Optional testing dataframe passed at instantiation. **features: python list** List of features to be processed passed at instantiation. **categorical: python list** List of categorical features found in the datasets: ["month", "Species", "ResultDir", "gps_cat", "sprayed", "Sunrise", "Sunset"]. **cat_features: python list** List of categorical features to be processed, dependent upon **features.** **ohe: object** sk-learn OneHotEncoder instance. **encoded_features: python list** List of features after one hot encoding. **encoded_test_features: python list** List of features of **test_df** after one hot encoding. **mode: str**

	Toggle selector to operate the class in either wnv mode or num_mosquitos mode. **y: pandas series** Training target variable. **X: pandas dataframe** Training predictors. **X_train: pandas dataframe** Training predictors after application of sk-learn's train-test-split. **X_test: pandas dataframe** Testing predictors derived from training dataset after application of sk-learn's train-test-split. **y_train: pandas dataframe** Training target after application of sk-learn's train-test-split. **y_test: pandas dataframe** Testing target derived from training dataset after application of sk-learn's train-test-split. **clean_executed: boolean** Flags whether the .clean() method has been executed. **num_mosquito_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **tmax_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.tmin_ss: object**

	sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.tavg_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.dewpoint_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.wetbulb_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.heat_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.cool_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.preciptotal_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.windspeed_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.pressure_ss: object** sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column. **self.sealevel_ss: object**

	sk-learn standard scaler object fitted to the relevant column in the training dataset, saved for transforming relevant testing data column.

Examples:

```
import pandas as pd
from data_cleaner_with_test_df_functionality import DataCleaner

features = ['Species', 'Tmax', 'Tmin', 'Tavg', 'DewPoint', 'WetBulb',
'Heat', 'Cool', 'PrecipTotal',
              'ResultSpeed', 'ResultDir', 'StnPressure', 'SeaLevel',
'gps_cat', 'sprayed', 'month']

df = pd.read_csv("df.csv", low_memory=False)
test_df = pd.read_csv("test_df.csv", low_memory=False)

cleaned = DataCleaner(df, features, test_df, mode="wnv")

cleaned.clean()
```

```
import pandas as pd
from data_cleaner_with_test_df_functionality import DataCleaner

features = ['Species', 'Tmax', 'Tmin', 'Tavg', 'DewPoint', 'WetBulb',
'Heat', 'Cool', 'PrecipTotal',
              'ResultSpeed', 'ResultDir', 'StnPressure', 'SeaLevel',
'gps_cat', 'sprayed', 'month']

df = pd.read_csv("df.csv", low_memory=False)

cleaned = DataCleaner(df, features)

cleaned.clean()
```

Methods

.clean()	Cleans training dataframe.
.clean_test()	Cleans testing dataframe. Must have executed .clean() prior to executing this method.