```python
import os
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist

import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv("data.csv")
genre_data = pd.read_csv("data_by_genres.csv")
year_data = pd.read_csv("data_by_year.csv")
```

```
----------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
Cell In[4], line 2
      1 data = pd.read_csv("data.csv")
----> 2 genre_data = pd.read_csv("data_by_genres.csv")
      3 year_data = pd.read_csv("data_by_year.csv")

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:211, in
deprecate_kwarg.<locals>._deprecate_kwarg.<locals>.wrapper(*args,
**kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:331, in
deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*arg
s, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327
msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
```

```
--> 331 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:950,
in read_csv(filepath_or_buffer, sep, delimiter, header, names,
index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine,
converters, true_values, false_values, skipinitialspace, skiprows,
skipfooter, nrows, na_values, keep_default_na, na_filter, verbose,
skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col,
date_parser, dayfirst, cache_dates, iterator, chunksize, compression,
thousands, decimal, lineterminator, quotechar, quoting, doublequote,
escapechar, comment, encoding, encoding_errors, dialect,
error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace,
low_memory, memory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:605,
in _read(filepath_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1442,
in TextFileReader.__init__(self, f, engine, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1735,
in TextFileReader._make_engine(self, f, engine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File ~\anaconda3\lib\site-packages\pandas\io\common.py:856, in
get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    851 elif isinstance(handle, str):
    852     # Check whether the filename is to be opened in binary
mode.
    853     # Binary mode does not support 'encoding' and 'newline'.
    854     if ioargs.encoding and "b" not in ioargs.mode:
    855         # Encoding
--> 856         handle = open(
    857             handle,
    858             ioargs.mode,
    859             encoding=ioargs.encoding,
    860             errors=errors,
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
    865         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory:
'data_by_genres.csv'
```

```python
from yellowbrick.target import FeatureCorrelation

feature_names = ['acousticness', 'danceability', 'energy',
'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'tempo',
'valence','duration_ms','explicit','key','mode','year']

X, y = data[feature_names], data['popularity']

# Create a list of the feature names
features = np.array(feature_names)

# Instantiate the visualizer
visualizer = FeatureCorrelation(labels=features)

plt.rcParams['figure.figsize']=(20,20)
visualizer.fit(X, y)     # Fit the data to the visualizer
visualizer.show()
```