laq3bp2om

September 10, 2024

```
[1]: # Import necessary libraries
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder
     from sklearn.linear_model import LogisticRegression
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import classification_report

     # Load the dataset
     data = pd.read_csv('voice.csv')

     # Encode the categorical labels
     label_encoder = LabelEncoder()
     data['label'] = label_encoder.fit_transform(data['label'])

     # Split the dataset into features and target variable
     X = data.drop(columns=['label'])
     y = data['label']

     # Split the dataset into training and testing sets (80% train, 20% test)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
      ↪random_state=42)

     # Logistic Regression model
     log_reg_model = LogisticRegression(max_iter=1000, random_state=42)
     log_reg_model.fit(X_train, y_train)

     # Predictions and evaluation for Logistic Regression
     y_pred_log_reg = log_reg_model.predict(X_test)
     print("Logistic Regression:\n", classification_report(y_test, y_pred_log_reg))

     # k-Nearest Neighbors (k-NN) model
     knn_model = KNeighborsClassifier()
     knn_model.fit(X_train, y_train)

     # Predictions and evaluation for k-NN
```

```python
y_pred_knn = knn_model.predict(X_test)
print("k-NN:\n", classification_report(y_test, y_pred_knn))

# Decision Tree model
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)

# Predictions and evaluation for Decision Tree
y_pred_tree = decision_tree_model.predict(X_test)
print("Decision Tree:\n", classification_report(y_test, y_pred_tree))
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[1], line 11
      8 from sklearn.metrics import classification_report
     10 # Load the dataset
---> 11 data = pd.read_csv('voice.csv')
     13 # Encode the categorical labels
     14 label_encoder = LabelEncoder()

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:948, in␣
 ↪read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col,␣
 ↪usecols, 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, date_format, dayfirst, cache_dates, iterator,␣
 ↪chunksize, compression, thousands, decimal, lineterminator, quotechar,␣
 ↪quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect ␣
 ↪on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision,␣
 ↪storage_options, dtype_backend)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (…)
    944     dtype_backend=dtype_backend,
    945 )
    946 kwds.update(kwds_defaults)
--> 948 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:611, in␣
 ↪_read(filepath_or_buffer, kwds)
    608 _validate_names(kwds.get("names", None))
    610 # Create the parser.
--> 611 parser = TextFileReader(filepath_or_buffer, **kwds)
    613 if chunksize or iterator:
    614     return parser

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1448, in␣
 ↪TextFileReader.__init__(self, f, engine, **kwds)
   1445     self.options["has_index_names"] = kwds["has_index_names"]
   1447 self.handles: IOHandles | None = None
-> 1448 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1705, in␣
 ↪TextFileReader._make_engine(self, f, engine)
   1703     if "b" not in mode:
   1704         mode += "b"
-> 1705 self.handles = get_handle(
   1706     f,
   1707     mode,
   1708     encoding=self.options.get("encoding", None),
   1709     compression=self.options.get("compression", None),
   1710     memory_map=self.options.get("memory_map", False),
   1711     is_text=is_text,
   1712     errors=self.options.get("encoding_errors", "strict"),
   1713     storage_options=self.options.get("storage_options", None),
   1714 )
   1715 assert self.handles is not None
   1716 f = self.handles.handle

File ~\anaconda3\Lib\site-packages\pandas\io\common.py:863, in␣
 ↪get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,␣
 ↪errors, storage_options)
    858 elif isinstance(handle, str):
    859     # Check whether the filename is to be opened in binary mode.
    860     # Binary mode does not support 'encoding' and 'newline'.
    861     if ioargs.encoding and "b" not in ioargs.mode:
    862         # Encoding
--> 863         handle = open(
    864             handle,
    865             ioargs.mode,
    866             encoding=ioargs.encoding,
    867             errors=errors,
    868             newline="",
    869         )
    870     else:
    871         # Binary mode
    872         handle = open(handle, ioargs.mode)

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

```
[ ]:
```