DonorsChoose Data

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of donating. Paper cuts, a large number of requests, and limited time is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as accurately as possible?
- How to ensure that teachers' project proposals are not allocated unnecessarily to a particular donor, donor preferences and donor geography is not taken into consideration for donors to see and fund appropriate projects?
- The goal of the competition is to provide donors with the most exciting projects to fund. A project proposal submitted by a teacher will be approved, posted to the website, and funded by a donor. However, not every project is equally exciting to donors and will remain fundraising for days and often expires (fundraising ends before they can fund). How can we predict which projects are most likely to attract donors quickly?

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`teacher_id`	A unique identifier for the proposal's teacher. *Ex: b2cb74a8dcd9d8f8ca7b0d0b187d999b*
`teacher_prefix`	The teacher's title. *Ex: Mrs.*, *Mr.*, *Ms.*, *Teacher*, *Dr.*
`school_state`	State where school is located (Chicago Public Schools). *Ex: WY*
`project_submitted_datetime`	Datetime when project was submitted. *Ex: 2016-04-27 00:27:36*
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: *Ex: Grades PreK-2* *Grades 3-5* *Grades 6-8* *Grades 9-12*
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: *Ex: Applied Sciences, Health & Life Science* *Care & Hygiene* *Character Education* *Civics & Government* *Climate, Sustainability* *Community Service* *Music & The Arts* *Environmental Science* *Extracurricular* *Character Education, Civics & Government, Community Service*
`school_subject_categories`	State where school is located (Chicago Public Schools). *Ex: Literacy & Language, Math & Science*
`project_resource_summary`	An explanation of the resources needed for the project. *Ex: My students need hands on literacy materials to manipulate sounds and letters to practice reading and spelling.*
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project was submitted. *Ex: 2016-04-27 00:27:36*
`teacher_number_of_previously_posted_projects`	Number of previously posted projects by the same teacher. *Ex: 2*

* Application essays are only for California, Chicago, New York City, and Texas.

The data set also contains the following fields for the outcome and additional projects:

Feature	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved project A's proposal. *Ex: for a project A, if project A's proposal was approved, then `project_is_approved` for project A is 1, otherwise it is 0*

Notes on the Essay Data

- Prior to May 17, 2016, the prompts for the essays were as follows:
  - `__ preparation` refers to the prompt for essay 1.
  - `__ your students` refers to the prompt for essay 2.
- After May 17, 2016, the prompts for the essays were as follows:
  - `__ your students to` refers to the prompt for essay 1.
  - `__ their lives` refers to the prompt for essay 2.
- Teachers can enter a maximum of 2 essays for projects in California and a maximum of 1 essay for the other 3 essays when changing the prompt.
- `__ your students` or `__ their lives` refers to the teacher's project proposal.
- `__ preparation` refers to the teacher's description of how their students will benefit from the project.

Notes on the Resource Data

- For all projects with `project_submitted_datetime` of 2016-05-17 and later, the values of `project_essay_3` and `project_essay_4` will be NULL.

File	Description
`resources.csv`	Provides information about the resources required for each project. Several rows per `project_id`.

Notes on the Essay Data

- Prior to May 17, 2016, the prompts for the essays were as follows:
  - `__ preparation` refers to the prompt for essay 1.
  - `__ your students` refers to the prompt for essay 2.
- After May 17, 2016, the prompts for the essays were as follows:
  - `__ your students to` refers to the prompt for essay 1.
  - `__ their lives` refers to the prompt for essay 2.
- Teachers can enter a maximum of 2 essays for projects in California and a maximum of 1 essay for the other 3 essays when changing the prompt.
- `__ your students` or `__ their lives` refers to the teacher's project proposal.
- `__ preparation` refers to the teacher's description of how their students will benefit from the project. Each time in this file represents a resource.

Feature	Description
`id`	A project id. Join resources.csv on `id` to `projects_data.csv`
`description`	Description of the resource. Example: *Tissue*, *Decorative Mirrors*, *Pen*, *LP*
`quantity`	Quantity of the resource required. Example: *3*
`price`	Price of the resource required. Example: *9.95*

Note: Some projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you can join to the other files.

Note: Resources are not to scale.

The files ending in `.zip` contain the full data set, and will be released on Kaggle.com.

File	Description
`sample_submission.csv`	A sample submission in the correct format for the competition. It assumes all projects are funded.

Notes on the Essay Data

- Prior to May 17, 2016, the prompts for the essays were as follows:
  - `__ preparation` refers to the prompt for essay 1.
  - `__ your students` refers to the prompt for essay 2.
- After May 17, 2016, the prompts for the essays were as follows:
  - `__ your students to` refers to the prompt for essay 1.
  - `__ their lives` refers to the prompt for essay 2.
- Teachers can enter a maximum of 2 essays for projects in California and a maximum of 1 essay for the other 3 essays when changing the prompt.
- `__ your students` or `__ their lives` refers to the teacher's project proposal.
- `__ preparation` refers to the teacher's description of how their students will benefit from the project.

1. Reading Data

```
In [1]: %matplotlib inline
        import os
        import sqlite3
        import pandas as pd
        import numpy
        import nltk
        import string
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.feature_extraction import TfidfVectorizer
        from sklearn.feature_extraction import CountVectorizer
        from sklearn.preprocessing import Normalizer
        from scipy.sparse.base import issparse
        from wordcloud import WordCloud, STOPWORDS
        from PyPDF2 import PdfFileReader
        import warnings
        warnings.filterwarnings("ignore")
        from TqdmJupyter.tqdm import tqdmjn
        import IPython.display as ipd
        import gensim
        import pickle

        from IPython.display import HTML

        HTML('''<script type="text/javascript">
        var kill_rows = document.querySelectors("div.dataframe tr:nth-child(n+1000)");
        for (var i = 0, len = kill_rows.length; i < len; i++) {
            kill_rows[i].parentNode.removeChild(kill_rows[i]);
        }
        console.log("Done!");
        </script>''')
```

```
Out[1]: Done!
```

```
In [2]: %cd /drive/MyDrive/Colab Notebooks/Drive_Stanford/DonorsChoose/Data
        !ls -l -h
```

```
        /drive/MyDrive/Colab Notebooks/Drive_Stanford/DonorsChoose/Data
        total 1.6G
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description.pdf
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description.txt
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description.zip
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description_Resources.pdf
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description_Resources.zip
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description_Resources.txt
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description.zip
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description.txt
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description_Resources.zip
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description_Resources.txt
        -rrr- 1 root root 1.9M Jan  1  2016 Data_Description.zip
```

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler, LabelEncoder
        from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, roc_curve
        import matplotlib.pyplot as plt
        import seaborn as sns
        from xgboost import XGBClassifier, plot_importance
        from lightgbm import LGBMClassifier
        from catboost import CatBoostClassifier
        from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.svm import SVC
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import cross_val_score
        from sklearn.feature_selection import SelectKBest, f_classif
        from sklearn.pipeline import Pipeline
        from sklearn.base import BaseEstimator, TransformerMixin
        from sklearn.preprocessing import PolynomialFeatures
        from scipy.stats import ttest_ind, chi2_contingency
        from statsmodels.stats.multitest import multipletests
        from sklearn.decomposition import PCA
        from sklearn.cluster import KMeans
        from sklearn.metrics import silhouette_score
        from sklearn.manifold import TSNE
        from sklearn.feature_selection import mutual_info_classif
        from sklearn.inspection import permutation_importance
        from sklearn.model_selment import StratifiedKFold
        from sklearn.metrics import (accuracy_score, precision_score, recall_score,
                                     f1_score, roc_auc_score, confusion_matrix,
                                     classification_report, roc_curve,
                                     precision_recall_curve, average_precision_score)
        from sklearn.calibration import CalibrationDisplay
        from sklearn.model_selection import (train_test_split, cross_val_score,
                                             StratifiedKFold, GridSearchCV,
                                             RandomizedSearchCV)
        from sklearn.preprocessing import (StandardScaler, MinMaxScaler,
                                           LabelEncoder, OneHotEncoder,
                                           PolynomialFeatures, PowerTransformer)
        from sklearn.feature_selection import (SelectKBest, f_classif,
                                               mutual_info_classif, RFE,
                                               SelectFromModel, VarianceThreshold)
        from sklearn.decomposition import PCA, TruncatedSVD
        from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
        from sklearn.manifold import TSNE, Isomap
        from sklearn.base import BaseEstimator, TransformerMixin
        from sklearn.pipeline import Pipeline, FeatureUnion
        from sklearn.ensemble import (RandomForestClassifier,
                                      GradientBoostingClassifier,
                                      AdaBoostClassifier, VotingClassifier,
                                      StackingClassifier)
        from sklearn.linear_model import LogisticRegression, SGDClassifier
        from sklearn.tree import DecisionTreeClassifier, plot_tree
        from sklearn.svm import SVC, LinearSVC
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import (confusion_matrix, classification_report,
                                     roc_curve, roc_auc_score, precision_recall_curve,
                                     average_precision_score, calibration_curve,
                                     ConfusionMatrixDisplay, RocCurveDisplay,
                                     PrecisionRecallDisplay)
        from sklearn.calibration import CalibrationDisplay
        from sklearn.inspection import permutation_importance, PartialDependence
        from sklearn.model_selection import (StratifiedKFold, GridSearchCV,
                                             RandomizedSearchCV, cross_val_score,
                                             learning_curve, validation_curve)
        from scipy.stats import (ttest_ind, chi2_contingency, mannwhitneyu,
                                 shapiro, kstest, pearsonr, spearmanr)
        from statsmodels.stats.multitest import multipletests, fdrcorrection
        from statsmodels.stats.proportion import proportion_confint
        import joblib, pickle, warnings, os, gc, time, logging, copy, pathlib
        from pathlib import Path
        from collections import Counter, defaultdict, OrderedDict
        from datetime import datetime, timedelta
        from tqdm.notebook import tqdm, trange
        from IPython.display import display, HTML
        pd.set_option('display.max_columns', 200)
        pd.set_option('display.max_rows', 200)
        pd.set_option('display.float_format', lambda x: f'{x:.4f}')
        np.random.seed(42)
        warnings.filterwarnings('ignore')
        %matplotlib inline
        %matplotlib notebook
        plt.rcParams['figure.dpi'] = 100
        plt.rcParams['figure.figure.max_open_warning'] = 0
        plt.rcParams['font.family'] = 'sans-serif'
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial']
        plt.rcParams['axes.unicode_minus'] = False
        sns.set_style('whitegrid')
        sns.set_palette('deep')
        sns.set_context('notebook')
        %load_ext autotime
        %load_ext pyinstrument
        %load_ext memory_profiler
        %load_ext line_profiler
        %load_ext rpy2.ipython
        %load_ext sql
        %load_ext cythonmagic
        %load_ext pyspark
        %load_ext ipyparallel
        %load_ext ipywidgets
        %load_ext ipyvolume
        %load_ext ipyleaflet
        %load_ext ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
        %load_ext ipyvolume.pyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume.ipyvolume
```

```
In [2]: df = pd.read_csv('credit_scoring.csv')
        print(f"Shape: {df.shape}")
        display(df.head(5))
        display(df.tail(5))
        df.info(memory_usage='deep')
        df.describe(percentile=[.01,.05,.10,.25,.50,.75,.90,.95,.99]).T.round(3)
        df.nunique(dropna=False).sort_values(ascending=False).head(40)
        df.isnull().sum().sort_values(ascending=False).head(40)
        df.duplicated(keep=False).sum()
        df.dtypes.value_counts()
        df.select_dtypes(include=['object','category','bool','datetime64[ns]','int8','int16','int32','int64','uint8','uint16','uint32','uint64','float16','float32','float64','complex64','complex128','Sparse','Period','Interval','boolean','Int8','Int16','Int32','Int64','UInt8','UInt16','UInt32','UInt64','Float32','Float64','Int64Dtype','UInt64Dtype','Float64Dtype','BooleanDtype','StringDtype','CategoricalDtype','DatetimeTZDtype','PeriodDtype','IntervalDtype','SparseDtype','ArrowDtype','NumpyEXTENSIONDtype','NumpyDtype','PandasDtype','ExtensionDtype','CategoricalIndexDtype','DatetimeIndexDtype','TimedeltaIndexDtype','PeriodIndexDtype','IntervalIndexDtype','MultiIndexDtype','RangeIndexDtype','DatetimeArrayDtype','TimedeltaArrayDtype','PeriodArrayDtype','IntervalArrayDtype','SparseArrayDtype','ArrowStringArrayDtype','NumpyStringDtype','PandasStringDtype','ExtensionArrayDtype','CategoricalDtypeType','DatetimeTZDtypeType','PeriodDtypeType','IntervalDtypeType','SparseDtypeType','ArrowDtypeType','NumpyEXTENSIONDtypeType','NumpyDtypeType','PandasDtypeType','ExtensionDtypeType','CategoricalIndexDtypeType','DatetimeIndexDtypeType','TimedeltaIndexDtypeType','PeriodIndexDtypeType','IntervalIndexDtypeType','MultiIndexDtypeType','RangeIndexDtypeType','DatetimeArrayDtypeType','TimedeltaArrayDtypeType','PeriodArrayDtypeType','IntervalArrayDtypeType','SparseArrayDtypeType','ArrowStringArrayDtypeType','NumpyStringDtypeType','PandasStringDtypeType','ExtensionArrayDtypeType','CategoricalDtypeTypeType','DatetimeTZDtypeTypeType','PeriodDtypeTypeType','IntervalDtypeTypeType','SparseDtypeTypeType','ArrowDtypeTypeType','NumpyEXTENSIONDtypeTypeType','NumpyDtypeTypeType','PandasDtypeTypeType','ExtensionDtypeTypeType','CategoricalIndexDtypeTypeType','DatetimeIndexDtypeTypeType','TimedeltaIndexDtypeTypeType','PeriodIndexDtypeTypeType','IntervalIndexDtypeTypeType','MultiIndexDtypeTypeType','RangeIndexDtypeTypeType','DatetimeArrayDtypeTypeType','TimedeltaArrayDtypeTypeType','PeriodArrayDtypeTypeType','IntervalArrayDtypeTypeType','SparseArrayDtypeTypeType','ArrowStringArrayDtypeTypeType','NumpyStringDtypeTypeType','PandasStringDtypeTypeType','ExtensionArrayDtypeTypeType','CategoricalDtypeTypeTypeType','DatetimeTZDtypeTypeTypeType','PeriodDtypeTypeTypeType','IntervalDtypeTypeTypeType','SparseDtypeTypeTypeType','ArrowDtypeTypeTypeType','NumpyEXTENSIONDtypeTypeTypeType','NumpyDtypeTypeTypeType','PandasDtypeTypeTypeType','ExtensionDtypeTypeTypeType','CategoricalIndexDtypeTypeTypeType','DatetimeIndexDtypeTypeTypeType','TimedeltaIndexDtypeTypeTypeType','PeriodIndexDtypeTypeTypeType','IntervalIndexDtypeTypeTypeType','MultiIndexDtypeTypeTypeType','RangeIndexDtypeTypeTypeType','DatetimeArrayDtypeTypeTypeType','TimedeltaArrayDtypeTypeTypeType','PeriodArrayDtypeTypeTypeType','IntervalArrayDtypeTypeTypeType','SparseArrayDtypeTypeTypeType','ArrowStringArrayDtypeTypeTypeType','NumpyStringDtypeTypeTypeType','PandasStringDtypeTypeTypeType','ExtensionArrayDtypeTypeTypeType','CategoricalDtypeTypeTypeTypeType','DatetimeTZDtypeTypeTypeTypeType','PeriodDtypeTypeTypeTypeType','IntervalDtypeTypeTypeTypeType','SparseDtypeTypeTypeTypeType','ArrowDtypeTypeTypeTypeType','NumpyEXTENSIONDtypeTypeTypeTypeType','NumpyDtypeTypeTypeTypeType','PandasDtypeTypeTypeTypeType','ExtensionDtypeTypeTypeTypeType','CategoricalIndexDtypeTypeTypeTypeType','DatetimeIndexDtypeTypeTypeTypeType','TimedeltaIndexDtypeTypeTypeTypeType','PeriodIndexDtypeTypeTypeTypeType','IntervalIndexDtypeTypeTypeTypeType','MultiIndexDtypeTypeTypeTypeType','RangeIndexDtypeTypeTypeTypeType','DatetimeArrayDtypeTypeTypeTypeType','TimedeltaArrayDtypeTypeTypeTypeType','PeriodArrayDtypeTypeTypeTypeType','IntervalArrayDtypeTypeTypeTypeType','SparseArrayDtypeTypeTypeTypeType','ArrowStringArrayDtypeTypeTypeTypeType','NumpyStringDtypeTypeTypeTypeType','PandasStringDtypeTypeTypeTypeType','ExtensionArrayDtypeTypeTypeTypeType','CategoricalDtypeTypeTypeTypeTypeType','DatetimeTZDtypeTypeTypeTypeTypeType','PeriodDtypeTypeTypeTypeTypeType','IntervalDtypeTypeTypeTypeTypeType','SparseDtypeTypeTypeTypeTypeType','ArrowDtypeTypeTypeTypeTypeType','NumpyEXTENSIONDtypeTypeTypeTypeTypeType','NumpyDtypeTypeTypeTypeTypeType','PandasDtypeTypeTypeTypeTypeType','ExtensionDtypeTypeTypeTypeTypeType','CategoricalIndexDtypeTypeTypeTypeTypeType','DatetimeIndexDtypeTypeTypeTypeTypeType','TimedeltaIndexDtypeTypeTypeTypeTypeType','PeriodIndexDtypeTypeTypeTypeTypeType','IntervalIndexDtypeTypeTypeTypeTypeType','MultiIndexDtypeTypeTypeTypeTypeType','RangeIndexDtypeTypeTypeTypeTypeType','DatetimeArrayDtypeTypeTypeTypeTypeType','TimedeltaArrayDtypeTypeTypeTypeTypeType','PeriodArrayDtypeTypeTypeTypeTypeType','IntervalArrayDtypeTypeTypeTypeTypeType','SparseArrayDtypeTypeTypeTypeTypeType','ArrowStringArrayDtypeTypeTypeTypeTypeType','NumpyStringDtypeTypeTypeTypeTypeType','PandasStringDtypeTypeTypeTypeTypeType','ExtensionArrayDtypeTypeTypeTypeTypeType','CategoricalDtypeTypeTypeTypeTypeTypeType','DatetimeTZDtypeTypeTypeTypeTypeTypeType','PeriodDtypeTypeTypeTypeTypeTypeType','IntervalDtypeTypeTypeTypeTypeTypeType','SparseDtypeTypeTypeTypeTypeTypeType','ArrowDtypeTypeTypeTypeTypeTypeType','NumpyEXTENSIONDtypeTypeTypeTypeTypeTypeType','NumpyDtypeTypeTypeTypeTypeTypeType','PandasDtypeTypeTypeTypeTypeTypeType','ExtensionDtypeTypeTypeTypeTypeTypeType','CategoricalIndexDtypeTypeTypeTypeTypeTypeType','DatetimeIndexDtypeTypeTypeTypeTypeTypeType','TimedeltaIndexDtypeTypeTypeTypeTypeTypeType','PeriodIndexDtypeTypeTypeTypeTypeTypeType','IntervalIndexDtypeTypeTypeTypeTypeTypeType','MultiIndexDtypeTypeTypeTypeTypeTypeType','RangeIndexDtypeTypeTypeTypeTypeTypeType','DatetimeArrayDtypeTypeTypeTypeTypeTypeType','TimedeltaArrayDtypeTypeTypeTypeTypeTypeType','PeriodArrayDtypeTypeTypeTypeTypeTypeType','IntervalArrayDtypeTypeTypeTypeTypeTypeType','SparseArrayDtypeTypeTypeTypeTypeTypeType','ArrowStringArrayDtypeTypeTypeTypeTypeTypeType','NumpyStringDtypeTypeTypeTypeTypeTypeType','PandasStringDtypeTypeTypeTypeTypeTypeType','ExtensionArrayDtypeTypeTypeTypeTypeTypeType','CategoricalDtypeTypeTypeTypeTypeTypeTypeType','DatetimeTZDtypeTypeTypeTypeTypeTypeTypeType','PeriodDtypeTypeTypeTypeTypeTypeTypeType','IntervalDtypeTypeTypeTypeTypeTypeTypeType','SparseDtypeTypeTypeTypeTypeTypeTypeType','ArrowDtypeTypeTypeTypeTypeTypeTypeType','NumpyEXTENSIONDtypeTypeTypeTypeTypeTypeTypeType','NumpyDtypeTypeTypeTypeTypeTypeTypeType','PandasDtypeTypeTypeTypeTypeTypeTypeType','ExtensionDtypeTypeTypeTypeTypeTypeTypeType','CategoricalIndexDtypeTypeTypeTypeTypeTypeTypeType','DatetimeIndexDtypeTypeTypeTypeTypeTypeTypeType','TimedeltaIndexDtypeTypeTypeTypeTypeTypeTypeType','PeriodIndexDtypeTypeTypeTypeTypeTypeTypeType','IntervalIndexDtypeTypeTypeTypeTypeTypeTypeType','MultiIndexDtypeTypeTypeTypeTypeTypeTypeType','RangeIndexDtypeTypeTypeTypeTypeTypeTypeType','DatetimeArrayDtypeTypeTypeTypeTypeTypeTypeType','TimedeltaArrayDtypeTypeTypeTypeTypeTypeTypeType','PeriodArrayDtypeTypeTypeTypeTypeTypeTypeType','IntervalArrayDtypeTypeTypeTypeTypeTypeTypeType','SparseArrayDtypeTypeTypeTypeTypeTypeTypeType','ArrowStringArrayDtypeTypeTypeTypeTypeTypeTypeType','NumpyStringDtypeTypeTypeTypeTypeTypeTypeType','PandasStringDtypeTypeTypeTypeTypeTypeTypeType','ExtensionArrayDtypeTypeTypeTypeTypeTypeTypeType','CategoricalDtypeTypeTypeTypeTypeTypeTypeTypeType','DatetimeTZDtypeTypeTypeTypeTypeTypeTypeTypeType','PeriodDtypeTypeTypeTypeTypeTypeTypeTypeType','IntervalDtypeTypeTypeTypeTypeTypeTypeTypeType','SparseDtypeTypeTypeTypeTypeTypeTypeTypeType','ArrowDtypeTypeTypeTypeTypeTypeTypeTypeType','NumpyEXTENSIONDtypeTypeTypeTypeTypeTypeTypeTypeType','NumpyDtypeTypeTypeTypeTypeTypeTypeTypeType','PandasDtypeTypeTypeTypeTypeTypeTypeTypeType','ExtensionDtypeTypeTypeTypeTypeTypeTypeTypeType','CategoricalIndexDtypeTypeTypeTypeTypeTypeTypeTypeType','DatetimeIndexDtypeTypeTypeTypeTypeTypeTypeTypeType','TimedeltaIndexDtypeTypeTypeTypeTypeTypeTypeTypeType','PeriodIndexDtypeTypeTypeTypeTypeTypeTypeTypeType','IntervalIndexDtypeTypeTypeTypeTypeTypeTypeTypeType','MultiIndexDtypeTypeTypeTypeTypeTypeTypeTypeType','RangeIndexDtypeTypeTypeTypeTypeTypeTypeTypeType','DatetimeArrayDtypeTypeTypeTypeTypeTypeTypeTypeType','TimedeltaArrayDtypeTypeTypeTypeTypeTypeTypeTypeType','PeriodArrayDtypeTypeTypeTypeTypeTypeTypeTypeType','IntervalArrayDtypeTypeTypeTypeTypeTypeTypeTypeType','SparseArrayDtypeTypeTypeTypeTypeTypeTypeTypeType','ArrowStringArrayDtypeTypeTypeTypeTypeTypeTypeTypeType','NumpyStringDtypeTypeTypeTypeTypeTypeTypeTypeType','PandasStringDtypeTypeTypeTypeTypeTypeTypeTypeType','ExtensionArrayDtypeTypeTypeTypeTypeTypeTypeTypeType','CategoricalDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','DatetimeTZDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','PeriodDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','IntervalDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','SparseDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','ArrowDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','NumpyEXTENSIONDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','NumpyDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','PandasDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','ExtensionDtypeTypeTypeTypeTypeTypeTypeTypeTypeType','CategoricalIndexDtypeTypeTypeTypeTypeTypeTypeTypeT
```

Vectorizing Sentiment score of the review: new_features/sentiment

```
# [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50]
# Converting the review into positive or negative sentiment
# [34]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [35]pip install textblob
for item in processed_review:
    try:
        blob = TextBlob(item)
        print(blob.sentiment)
        # print(blob.words)
        # print(blob.sentiment.polarity)
        # print(blob.sentiment.polarity, end=' ')
        # print(blob.sentiment.polarity, end=' ', file=open('output.txt', 'a'))
        polarity = blob.sentiment.polarity
        if polarity > 0:
            print('positive')
            # print('positive', file=open('output.txt', 'a'))
        elif polarity == 0:
            print('neutral')
            # print('neutral', file=open('output.txt', 'a'))
        else:
            print('negative')
            # print('negative', file=open('output.txt', 'a'))
    except:
        print('error')
        pass
# [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50]
# [36]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [37]pip install textblob
# [38]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [39]pip install textblob
# [40]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [41]pip install textblob
# [42]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [43]pip install textblob
# [44]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [45]pip install textblob
# [46]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [47]pip install textblob
# [48]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [49]pip install textblob
# [50]Sentiment analysis: TextBlob library(Natural Language Processing) is a python library for operating with text data.
# [50]pip install textblob
```

```
Sentiment(polarity=0.0, subjectivity=0.0)
neutral
Sentiment(polarity=0.0, subjectivity=0.0)
neutral
Sentiment(polarity=0.0, subjectivity=0.0)
neutral
```

```
In [1]# average accuracy
# regular average accuracy for each classes
# for each classes the average accuracy for each cross-validation is stored in this list
# the average accuracy for each classes for each cross-validation
average = np.zeros(shape=[7,n_cv])# 7 is each classes and 10 cv folds
# for each classes and each cross-validation the average accuracy
for i in range(0,7):
    # for each cross-validation
    for j in range(0,n_cv):
        # for each class
        average[i,j] = cm[i,i]/np.sum(cm[i,:])
print(average)
print("Average Accuracy",np.mean(average,axis=1))
print("Standard Deviation",np.std(average,axis=1))
print("Standard Error",np.std(average,axis=1)/np.sqrt(n_cv))
```

```
OUTPUT:
[[ 0.        0.        0.        
```

In [10]: # Scope: https://www.kaggle.com/bertcarremans/predicting-water-potability/data
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score

np.random.seed(42)
strat_train_set, strat_test_set = \
    *train*.iloc[train_index], *test*.iloc[test_index] \
    for train_index, test_index in \
    StratifiedShuffleSplit(n_splits=1, test_size=0.2, \
                           random_state=42).split(*train*, *train*["is_ave"])

cat_train_train_set, y_train1 = \
    strat_train_set.drop("is_ave", axis=1), \
    strat_train_set["is_ave"].copy()

*train_ave*.shape*train_train_set*.shape
*train_ave*.value_counts()/*train_train_set*.shape
*train_ave*.value_counts()/len(*train*)
*train_train_set*.shape*y_train1*.shape
*train_train_set*.info()*y_train1*.info()

pd.plotparam("figsize", (8,6))
*train_train_set*["ph"].hist(bins=50, alpha=0.5)
*train_ave*[*train_ave*["ph"]==1].["ph"].hist(bins=50, alpha=0.5)
plt.legend(["Non-potable water", "Potable water"])
plt.xlabel("ph")
plt.ylabel("Count")
plt.title("Water parameters vs ph")
plt.show()
pd.plotparam("figsize", (8,6))
*train*.hist()
```

```
array([[<AxisSubplot:>, <AxisSubplot:>, <AxisSubplot:>,
        
```