Report Midterm Exam

<u>Name'students</u>: Vũ Thị Quế Anh – 20070900
Nguyễn Thu Huyền – 20070937
<u>Lecture</u>: Trần Thị Oanh

Google Colaboratory

1. Cleaning dataset

Our team used python as a tool to process the data, drop unnecessary attributes, one-hot encoding to convert item data to numeric, and return NULL values to -1. The following is the data cleaning procedure.

a. Analysis

The team will first analyze the data set:

→ Display information on the first 5 observations.

```
[ ] cat_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks Data mining/Midterm exam/2022_shelter_cat_outcomes-train.csv')
    cat_df.head()
```

	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype	...	outcome_weekday	outcome_hour
0	2 weeks	A684346	Cat	domestic shorthair	orange	7/7/2014 0:00	7/22/2014 16:04	2014-07-22T16:04:00	NaN	Partner	...	Tuesday	16
1	1 month	A685067	Cat	domestic shorthair	blue /white	6/16/2014 0:00	8/14/2014 18:45	2014-08-14T18:45:00	Lucy	NaN	...	Thursday	18
2	3 months	A678580	Cat	domestic shorthair	white/black	3/26/2014 0:00	6/29/2014 17:45	2014-06-29T17:45:00	*Frida	Offsite	...	Sunday	17
3	1 year	A675405	Cat	domestic mediumhair	black/white	3/27/2013 0:00	3/28/2014 14:55	2014-03-28T14:55:00	Stella Luna	NaN	...	Friday	14
4	3 weeks	A670420	Cat	domestic shorthair	black/white	12/16/2013 0:00	1/9/2014 19:29	2014-01-09T19:29:00	NaN	Partner	...	Thursday	19

5 rows × 37 columns

→ Specifies the format and number of not-nulls for each field.

```
 0   age_upon_outcome       23536 non-null  object
 1   animal_id              23536 non-null  object
 2   animal_type            23536 non-null  object
 3   breed                  23536 non-null  object
 4   color                  20660 non-null  object
 5   date_of_birth          23536 non-null  object
 6   datetime               23536 non-null  object
 7   monthyear              23536 non-null  object
 8   name                   13331 non-null  object
 9   outcome_subtype        14866 non-null  object
 10  outcome_type           23534 non-null  object
 11  sex_upon_outcome       23536 non-null  object
 12  count                  23536 non-null  int64
 13  sex                    23536 non-null  object
 14  Spay/Neuter            23536 non-null  object
 15  Periods                23536 non-null  int64
 16  Period Range           23536 non-null  int64
 17  outcome_age_(days)     23536 non-null  int64
 18  outcome_age_(years)    23536 non-null  float64
 19  Cat/Kitten (outcome)   23536 non-null  object
 20  sex_age_outcome        23536 non-null  object
 21  age_group              23536 non-null  object
 22  dob_year               23536 non-null  int64
 23  dob_month              23536 non-null  int64
 24  dob_monthyear          23536 non-null  object
 25  outcome_month          23536 non-null  int64
 26  outcome_year           23536 non-null  int64
 27  outcome_weekday        23536 non-null  object
 28  outcome_hour           23536 non-null  int64
 29  breed1                 23536 non-null  object
 30  breed2                 39 non-null     object
 31  cfa_breed              23536 non-null  bool
 32  domestic_breed         23536 non-null  bool
 33  coat_pattern           15367 non-null  object
 34  color1                 23536 non-null  object
 35  color2                 8364 non-null   object
 36  coat                   23536 non-null  object
dtypes: bool(2), float64(1), int64(9), object(25)
memory usage: 6.3+ MB
```

→ Identify the columns present in the dataset.

```
[ ]  cat_df.columns
```

```
Index(['age_upon_outcome', 'animal_id', 'animal_type', 'breed', 'color',
       'date_of_birth', 'datetime', 'monthyear', 'name', 'outcome_subtype',
       'outcome_type', 'sex_upon_outcome', 'count', 'sex', 'Spay/Neuter',
       'Periods', 'Period Range', 'outcome_age_(days)', 'outcome_age_(years)',
       'Cat/Kitten (outcome)', 'sex_age_outcome', 'age_group', 'dob_year',
       'dob_month', 'dob_monthyear', 'outcome_month', 'outcome_year',
       'outcome_weekday', 'outcome_hour', 'breed1', 'breed2', 'cfa_breed',
       'domestic_breed', 'coat_pattern', 'color1', 'color2', 'coat'],
      dtype='object')
```

b. Plot

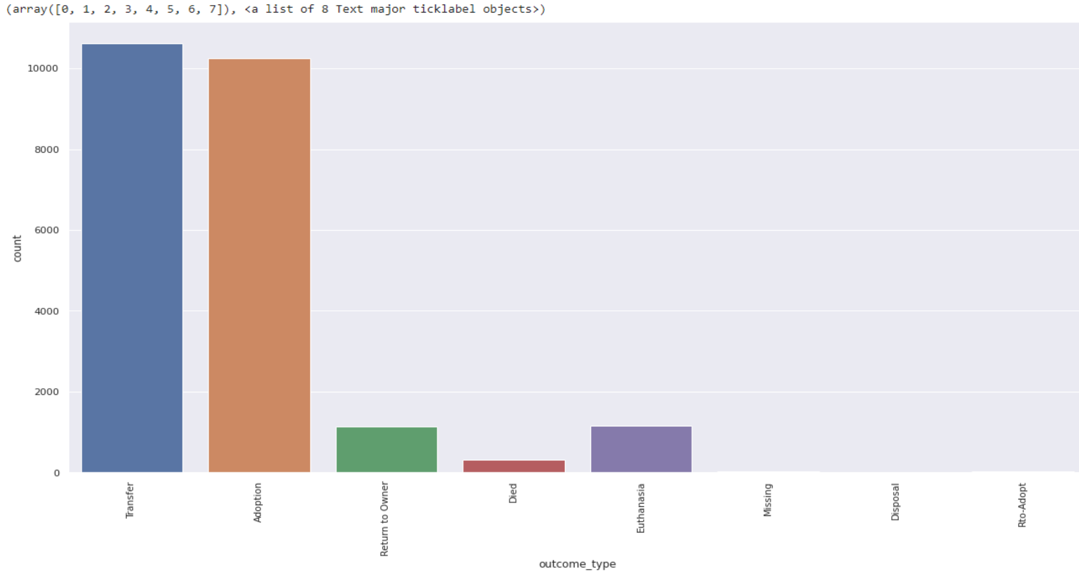
Visualize the data for statistics of the values contained in the data fields.

```
sns.countplot(x="breed", data=cat_df)
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59]),
 <a list of 60 Text major ticklabel objects>)
```


```
sns.countplot(x="outcome_type", data=cat_df)
plt.xticks(rotation=90)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7]), <a list of 8 Text major ticklabel objects>)
```



c. Clean data

After learning the information from the data fields, our team will remove the columns that completely duplicate the meaning.

```
cat_df = cat_df.drop(["age_upon_outcome", "animal_id", "animal_type", "color", "breed",
                      "date_of_birth", "datetime", "dob_monthyear", "monthyear", "outcome_subtype",
                      "sex", "Spay/Neuter",
                      "count", "Periods", "Period Range", "outcome_age_(days)", "outcome_age_(years)",
                      "sex_age_outcome",
                      "dob_year", "outcome_year"], axis=1)
```

The next step is to delete the duplicate rows.

```
cat_df = cat_df.drop_duplicates()
cat_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22452 entries, 0 to 23535
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   name                  13324 non-null  object
 1   outcome_type          22450 non-null  object
 2   sex_upon_outcome      22452 non-null  object
 3   Cat/Kitten (outcome)  22452 non-null  object
 4   age_group             22452 non-null  object
 5   dob_month             22452 non-null  int64
 6   outcome_month         22452 non-null  int64
 7   outcome_weekday       22452 non-null  object
 8   outcome_hour          22452 non-null  int64
 9   breed1                22452 non-null  object
 10  breed2                39 non-null     object
 11  cfa_breed             22452 non-null  bool
 12  domestic_breed        22452 non-null  bool
 13  coat_pattern          14659 non-null  object
 14  color1                22452 non-null  object
 15  color2                8095 non-null   object
 16  coat                  22452 non-null  object
dtypes: bool(2), int64(3), object(12)
memory usage: 2.8+ MB
```

d. Preprocessing

Our team applies one-hot encoding to bring the item data into digital form. In this encoding, each category value will correspond to a one-hot vector with k elements (k being the number of different values).

```
[ ] from sklearn.preprocessing import OneHotEncoder as OE
```

```
[ ] def onehot(x, codedict):
      if x is None or str(x) == "Unknown" or str(x).strip() == "" or str(x) == "nan":
        return None
      i = codedict.index(x)
      if i < 0 : return None
      else : return i
```

onehot name

```
[ ] #@title onehot name
    cat_df["name"] = cat_df["name"].apply(lambda x: 1 if x is None else 0)
```

onehot sex_upon_outcome

```
[ ] #@title onehot sex_upon_outcome
    Suo = list(cat_df["sex_upon_outcome"].unique())
    cat_df["sex_upon_outcome"] = cat_df['sex_upon_outcome'].apply(lambda x: onehot(x, Suo))
    Suo
```

```
['Intact Male', 'Intact Female', 'Spayed Female', 'Unknown', 'Neutered Male']
```

e. Missing data

With missing data, our team will fill those positions with the value -1.

```
[ ]  cat_df = cat_df.fillna(-1)
```

f. Save new dataset file

After processing, we see that all data fields have been filled, including 17 columns and 22452 records.

```
[ ]  cat_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 22452 entries, 0 to 23535
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   name                 22452 non-null  int64
 1   outcome_type         22452 non-null  object
 2   sex_upon_outcome     22452 non-null  float64
 3   Cat/Kitten (outcome) 22452 non-null  int64
 4   age_group            22452 non-null  int64
 5   dob_month            22452 non-null  int64
 6   outcome_month        22452 non-null  int64
 7   outcome_weekday      22452 non-null  int64
 8   outcome_hour         22452 non-null  int64
 9   breed1               22452 non-null  int64
 10  breed2               22452 non-null  float64
 11  cfa_breed            22452 non-null  bool
 12  domestic_breed       22452 non-null  bool
 13  coat_pattern         22452 non-null  float64
 14  color1               22452 non-null  int64
 15  color2               22452 non-null  float64
 16  coat                 22452 non-null  int64
dtypes: bool(2), float64(4), int64(10), object(1)
memory usage: 2.8+ MB
```

For the convenience of training and testing the model, we split the processed dataset into a file named: " refined_data.csv"

2. Predict output using AutoGluon

In this problem, we choose AutoGluon for predictive implementation. AutoGluon is a new open source AutoML library that automates deep learning (DL) and machine learning (ML) for real world applications involving image, text and tabular datasets. All in all, the AutoML model aims to automate all the time-consuming operations like algorithm selection, coding, pipeline development, and hyperparameter tuning, thus allowing data scientists to be more focused on rapidly addressing current business challenges. Within the pipeline, the AutoML framework:

- Considers and selects multiple machine learning algorithms from the available ones like the random forest, k-Nearest Neighbor, SVMs, etc.
- Performs data preprocessing steps like missing value imputation, feature scaling, feature selection, etc..,
- Optimization or the hyperparameter tuning for all of the models
- Decides/Tries multiple ways to ensemble or stack the algorithms

The following are the benefits of using the 'AutoGluon' library:

- Simplicity: Training of classification and regression models and deployment can be achieved with a few lines of code.
- Robustness: Without doing any feature engineering or data manipulation, users should be able to use raw data.
- Predictable-timing: Getting the best model under a specified time constraint.
- Fault-tolerance: Training can be resumed even if interrupted and the users can inspect all the intermediate steps.

Firstly, we need to install autogluon

```
!pip3 install torch==1.12+cpu torchvision==0.13.0+cpu torchtext==0.13.0 -f https://download.pytorch.org/whl/cpu/torch_stable.html

!pip3 install autogluon
```

a. Fit in Gluon

Our goal is to build a predictive model for predicting the outcome result of the cat recorded in the "outcome_type" attribute.

We use the dataset named "refinded_data.csv", and divide the dataset into train and test sets, with test_size = 0.1, random_state = 10

```
label = "outcome_type"
X, y = cat_df.drop(label, axis=1), cat_df[label]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=10)
X_train, y_train
```

Then, we use AutoGluon to train multiple models

Via a simple fit() call, AutoGluon can produce highly-accurate models to predict the values in one column of a data table based on the rest of the columns' values.

```
predictor = TabularPredictor(label="outcome_type", path=save_path)
```

```
predictor.fit(pd.concat([X_train, y_train], axis=1), time_limit=180, presets='best_quality')
# predictor.fit(pd.concat([X, y], axis=1))
```

The results show the accuracy of many models such as: KNeighborsUnif_BAG_L1, KNeighborsDist_BAG_L1, WeightedEnsemble_L2,....

b. Evaluated

In this part, we evaluate the model.

```
perf = predictor.evaluate_predictions(y_true=y_test, y_pred=y_pred, auxiliary_metrics=True)
```

```
Evaluation: accuracy on test data: 0.7622439893143366
Evaluations on test data:
{
    "accuracy": 0.7622439893143366,
    "balanced_accuracy": 0.23277175910729359,
    "mcc": 0.5871299880342093
}
```

The accuracy on test data is 0.76224

```
predictor.leaderboard(pd.concat([X_test, y_test], axis=1), silent=True)
```

	model	score_test	score_val	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal	pred_time_val_marginal	fit_time_marginal
0	WeightedEnsemble_L2	0.762244	0.764415	0.887889	2.669788	104.754212	0.008868	0.002897	2.473738
1	NeuralNetFastAI_BAG_L2	0.762244	0.764563	1.848261	6.824047	167.652690	0.703762	0.810019	65.275450
2	WeightedEnsemble_L3	0.762244	0.764563	1.852336	6.830050	167.692854	0.004074	0.006003	0.040164
3	NeuralNetFastAI_BAG_L1	0.760463	0.764019	0.608356	0.669309	102.198744	0.608356	0.669309	102.198744
4	KNeighborsDist_BAG_L1	0.708816	0.715318	0.270665	1.997582	0.081729	0.270665	1.997582	0.081729
5	KNeighborsUnif_BAG_L1	0.699466	0.710715	0.265478	3.347136	0.096766	0.265478	3.347136	0.096766

Above are 6 models of the training process, in which WeightedEnsembe_L3 is the best model.

```
predictor.get_model_best()
```

```
'WeightedEnsemble_L3'
```

We also predict some features' importance, and this shows that "sex_upon_outcome" is the most important feature.

- weights of features, the larger weights, the higher importance

```
#@title weights of features, the larger weights, the higher importance
predictor.feature_importance(pd.concat([X_test, y_test], axis=1))
```

```
These features in provided data are not utilized by the predictor and will be ignor
Computing feature importance via permutation shuffling for 13 features using 2245 r
        112.18s = Expected runtime (22.44s per shuffle set)
        79.16s  = Actual runtime (Completed 5 of 5 shuffle sets)
```

	importance	stddev	p_value	n	p99_high	p99_low
sex_upon_outcome	0.210958	0.004256	1.987194e-08	5	0.219721	0.202194
outcome_month	0.046503	0.006752	5.187620e-05	5	0.060407	0.032600
outcome_hour	0.042494	0.003143	3.566848e-06	5	0.048967	0.036022
dob_month	0.040356	0.005143	3.097299e-05	5	0.050945	0.029767
Cat/Kitten (outcome)	0.024766	0.003267	3.551695e-05	5	0.031493	0.018039
age_group	0.010245	0.000704	2.663289e-06	5	0.011695	0.008795
outcome_weekday	0.009889	0.004481	3.923632e-03	5	0.019115	0.000662
color1	0.001336	0.002500	1.490074e-01	5	0.006484	-0.003811
breed1	0.001158	0.002007	1.332322e-01	5	0.005290	-0.002974
coat_pattern	0.000713	0.001957	2.305661e-01	5	0.004742	-0.003317
coat	0.000445	0.001575	2.807190e-01	5	0.003688	-0.002797
breed2	0.000000	0.000000	5.000000e-01	5	0.000000	0.000000
color2	-0.002316	0.001649	9.826002e-01	5	0.001078	-0.005711

3. Tuning Hyper-parameters

In this part, people can use their testset for testing

```
save_path = "/content/drive/MyDrive/colab/Data mining/Midterm exam/models-optimized"
training_dataset_path = "/content/drive/MyDrive/colab/Data mining/Midterm exam/refined_data.csv"
testing_dataset_path = "" #put a path to your test dataset in here
```

a. Load data into a new format.
b. Search hyper-parameters
c. Test with real data (For teacher)