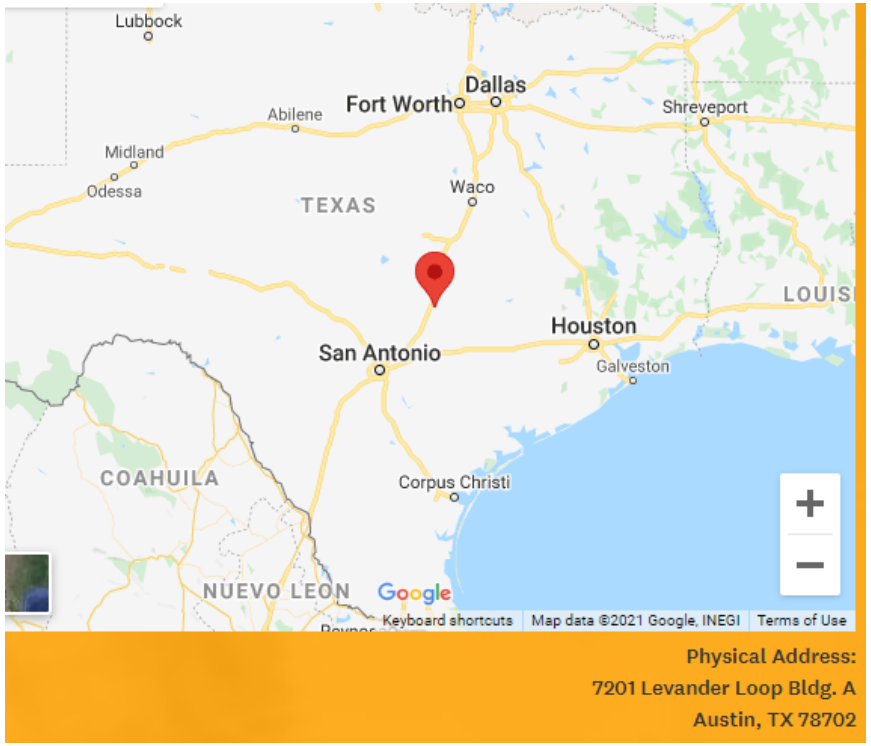
잃어버린 애완동물 미래 예측

2016108263 김한승

목차

- 데이터 소개
- 코드 리뷰
- Q&A

데이터 소개



미국 텍사스 주 휴스턴 시 왼편에 위치한 Austin Animal Center에서 제공하는 데이터로 이 동물 센터에서 보호하는, 보호하던 동물에 대한 데이터이다.

코드리뷰

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import sklearn
import sklearn.preprocessing
import sklearn.model_selection
import tensorflow as tf
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
```

데이터 시각화를 위한 모듈과 머신 러닝을 위한 모듈 추가

데이터 시각화를 위한 모듈

Numpy, pandas, seaborn, matplotlib.pypolt

머신 러닝을 위한 모듈

Sklearn, tensorflow

그 외 데이터 처리를 위한 모듈

datetime

예측을 위해 사용한 기술

- 텐서플로우와 로지스틱 회귀 모듈을 이용한 예측

코드리뷰

```
df = pd.read_csv('../input/train.csv')
df.head()
```

데이터 읽어오고 잘 읽어 왔는지 확인을 위해 데이터 앞부분 확인

	AnimalID	Name	DateTime	OutcomeType	OutcomeSubtype	AnimalType	SexuponOutcome	AgeuponOutcome	Breed	Color
0	A671945	Hambone	2014-02-12 18:22:00	Return_to_owner	NaN	Dog	Neutered Male	1 year	Shetland Sheepdog Mix	Brown/White
1	A656520	Emily	2013-10-13 12:44:00	Euthanasia	Suffering	Cat	Spayed Female	1 year	Domestic Shorthair Mix	Cream Tabby
2	A686464	Pearce	2015-01-31 12:28:00	Adoption	Foster	Dog	Neutered Male	2 years	Pit Bull Mix	Blue/White
3	A683430	NaN	2014-07-11 19:09:00	Transfer	Partner	Cat	Intact Male	3 weeks	Domestic Shorthair Mix	Blue Cream
4	A667013	NaN	2013-11-15 12:52:00	Transfer	Partner	Dog	Neutered Male	2 years	Lhasa Apso/Miniature Poodle	Tan

코드리뷰

```
features = df.drop(['AnimalID', 'OutcomeSubtype', 'Name'],axis=1).copy()
labels = df[['OutcomeType']].copy()
features.head()
```

데이터 중 분석에 사용되지 않을 데이터 요소 드랍

동물 개인 아이디, 동물 이름(품종X)

	DateTime	OutcomeType	AnimalType	SexuponOutcome	AgeuponOutcome	Breed	Color
0	2014-02-12 18:22:00	Return_to_owner	Dog	Neutered Male	1 year	Shetland Sheepdog Mix	Brown/White
1	2013-10-13 12:44:00	Euthanasia	Cat	Spayed Female	1 year	Domestic Shorthair Mix	Cream Tabby
2	2015-01-31 12:28:00	Adoption	Dog	Neutered Male	2 years	Pit Bull Mix	Blue/White
3	2014-07-11 19:09:00	Transfer	Cat	Intact Male	3 weeks	Domestic Shorthair Mix	Blue Cream
4	2013-11-15 12:52:00	Transfer	Dog	Neutered Male	2 years	Lhasa Apso/Miniature Poodle	Tan

코드리뷰

```
def mapDateTime(row):
    try:
        dt = datetime.strptime(row['DateTime'], '%Y-%m-%d %H:%M:%S')
        return pd.Series([dt.year, dt.month, dt.day], index=['year', 'month', 'day'])
    except:
        return pd.Series([2015, 1, 2], index=['year', 'month', 'day'])

def mapSexuponOutcome(row):
    try:
        intactness, gender = row['SexuponOutcome'].split(' ')
        intactness_val = 1
        gender_val = 1
        if intactness in ('Neutered', 'Spayed', 'neutered', 'spayed'):
            intactness_val = 0
        if gender in ('female', 'Female'):
            gender_val = 0
        return pd.Series([intactness_val, 1-intactness_val, 1-gender_val, gender_val], index=['intact', 'notintact', 'female', 'mal
e'])
    except:
        return pd.Series([0, 1, 0, 1], index=['intact', 'notintact', 'female', 'male'])
```

```
def mapAgeuponOutcome(row):
    try:
        digit, unit = row['AgeuponOutcome'].split(' ')
        unitDict = {'day':1, 'days':1, 'week':7, 'weeks':7, 'month':30, 'months':30, 'year':365, 'years':365}
        return pd.Series([int(digit)*unitDict[unit]], index=['age'])
    except:
        return pd.Series([0], index=['age'])

def mapBreed(row):
    try:
        breeds = row['Breed'].replace(' Mix', '').split('/')
        return pd.Series([breeds[0], breeds[-1]], index=['breed1', 'breed2'])
    except:
        return pd.Series(['Domestic Shorthair', 'Domestic Shorthair'], index=['breed1', 'breed2'])

def mapColor(row):
    try:
        colors = row['Color'].split('/')
        return pd.Series([colors[0], colors[-1]], index=['color1', 'color2'])
    except:
        return pd.Series(['Brown', 'Brown'], index=['color1', 'color2'])
```

데이터들을 날짜별로 나누고,
성별로 나누고,
몇 살때 잃어버렸는 지로 나누고,
털색으로 나누었다.

코드리뷰

```
datetimeDf = features.apply(mapDateTime, axis=1)
sexuponOutcomeDf = features.apply(mapSexuponOutcome, axis=1)
ageuponOutcomeDf = features.apply(mapAgeuponOutcome, axis=1)
breedLabelDf = features.apply(mapBreed, axis=1)
colorLabelDf = features.apply(mapColor, axis=1)
```

```
color1Df = pd.DataFrame(colorEncoder.transform(colorLabelDf['color1']))
color1Df.columns = colorEncoder.classes_
color1Df = color1Df.add_prefix('color1_')
color2Df = pd.DataFrame(colorEncoder.transform(colorLabelDf['color2']))
color2Df.columns = colorEncoder.classes_
color2Df = color2Df.add_prefix('color2_')
breed1Df = pd.DataFrame(breedEncoder.transform(breedLabelDf['breed1']))
breed1Df.columns = breedEncoder.classes_
breed1Df = breed1Df.add_prefix('breed1_')
breed2Df = pd.DataFrame(breedEncoder.transform(breedLabelDf['breed2']))
breed2Df.columns = breedEncoder.classes_
breed2Df = breed2Df.add_prefix('breed2_')
animalTypeDf = pd.DataFrame(animalTypeEncoder.transform(features['AnimalType']))
animalTypeDf.columns = animalTypeEncoder.classes_[[1]]
animalTypeDf = animalTypeDf.add_prefix('type_')
```

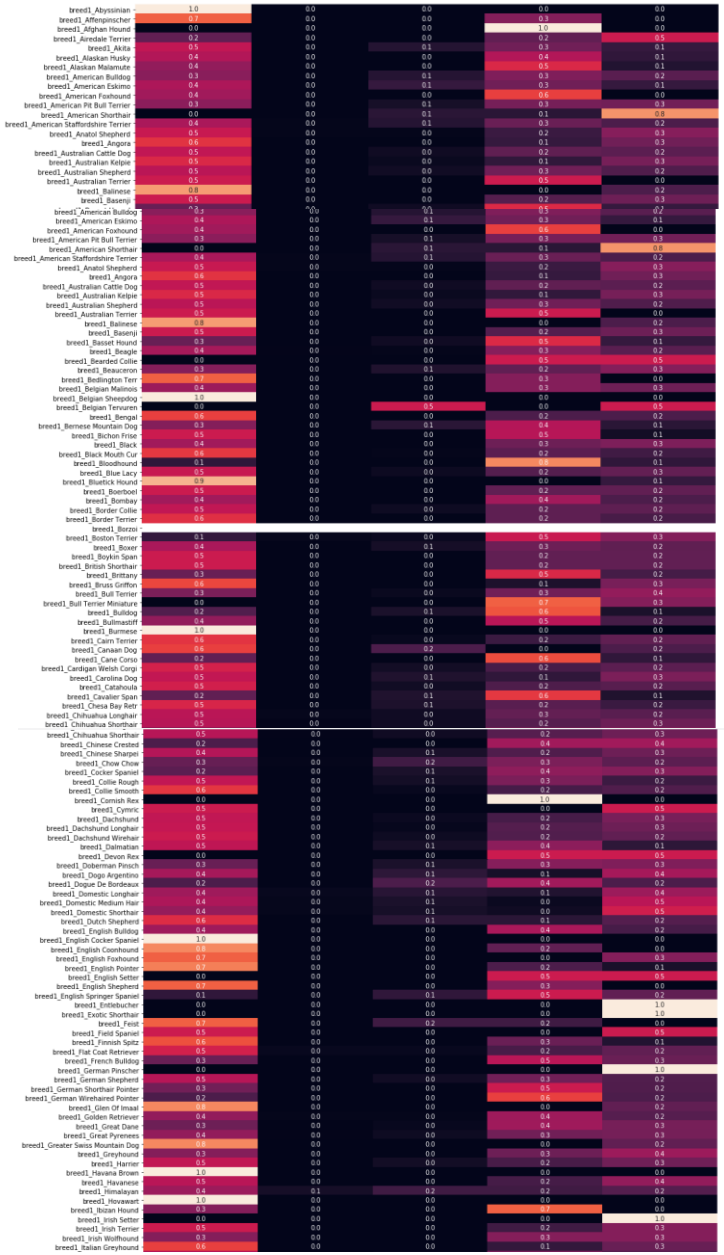
```
breedEncoder = sklearn.preprocessing.LabelBinarizer()
breedEncoder.fit(pd.concat([breedLabelDf['breed1'], breedLabelDf['breed2']]))
animalTypeEncoder = sklearn.preprocessing.LabelBinarizer()
animalTypeEncoder.fit(features['AnimalType'])
colorEncoder = sklearn.preprocessing.LabelBinarizer()
colorEncoder.fit(pd.concat([colorLabelDf['color1'], colorLabelDf['color2']]))
labelEncoder = sklearn.preprocessing.LabelEncoder()
labelEncoder.fit(labels)
```

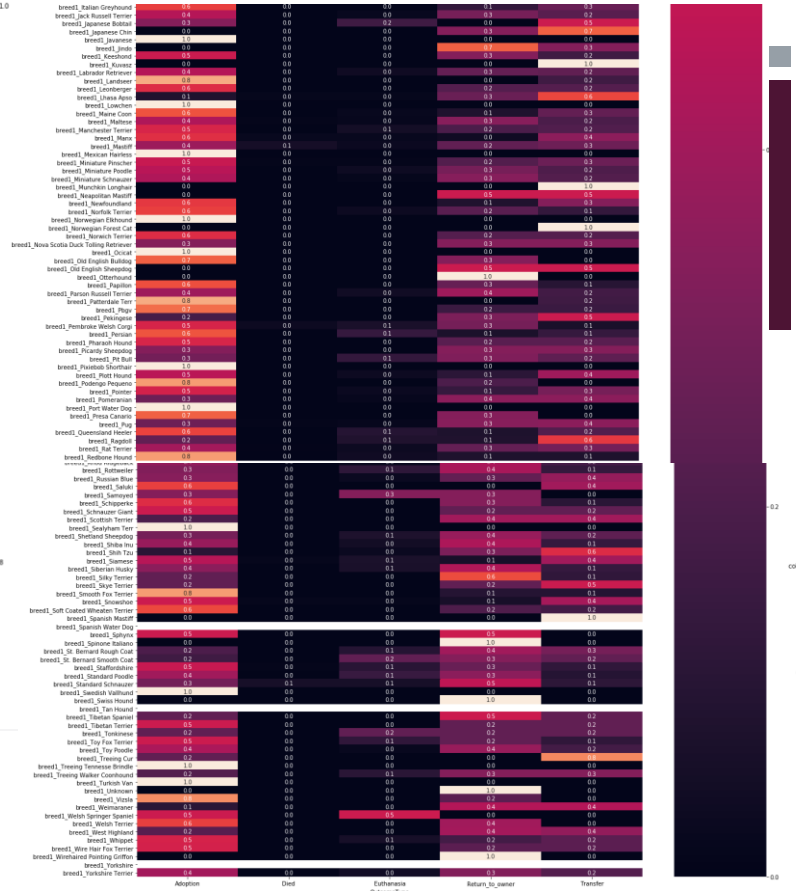
카테고리로 나뉘어 있던 자연어로 되어있던
데이터를 수치형 데이터로 변환 하였다.

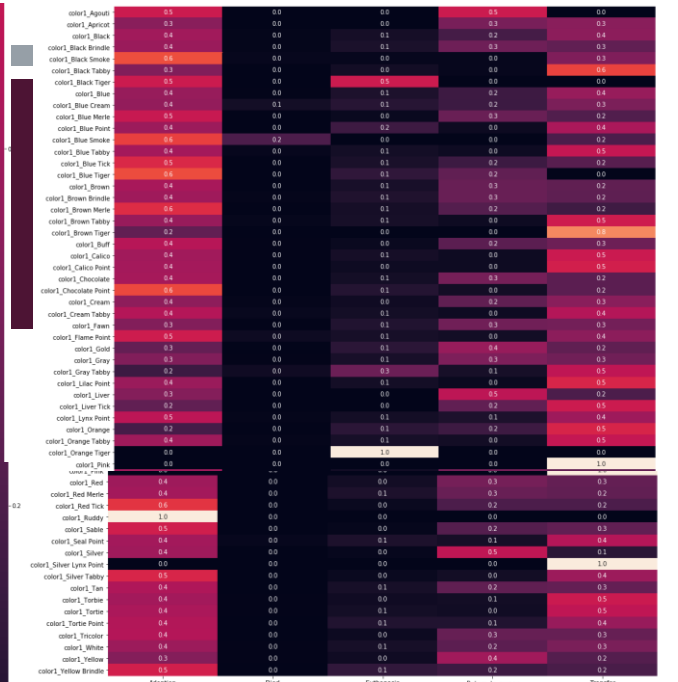
코드리뷰

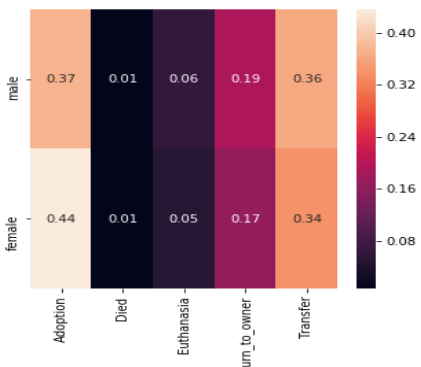
```
plt.subplots(figsize=(5,5))
labels.groupby('OutcomeType').size().plot.bar();

plt.subplots(figsize=(20,20))
tmp = pd.concat([color1Df, labels], axis=1).groupby('OutcomeType').sum()
tmp = tmp / tmp.sum()
sns.heatmap(tmp.T, annot=True, fmt='0.1f');

plt.subplots(figsize=(20,60))
tmp = pd.concat([breed1Df, labels], axis=1).groupby('OutcomeType').sum()
tmp = tmp / tmp.sum()
sns.heatmap(tmp.T, annot=True, fmt='0.1f');

plt.subplots(figsize=(5,5))
tmp = pd.concat([sexuponOutcomeDf[['intact', 'notintact']], labels], axis=1).groupby('OutcomeType').sum()
tmp = tmp / tmp.sum()
sns.heatmap(tmp.T, annot=True, fmt='0.2f');

plt.subplots(figsize=(5,5))
tmp = pd.concat([sexuponOutcomeDf[['male', 'female']], labels], axis=1).groupby('OutcomeType').sum()
tmp = tmp / tmp.sum()
sns.heatmap(tmp.T, annot=True, fmt='0.2f');

plt.subplots(figsize=(5,5))
tmp = pd.concat([animalTypeDf, 1-animalTypeDf.rename({'type_Dog': 'type_Cat','type_Cat': 'type_Dog'}, axis=1), labels], axis=1).group
by('OutcomeType').sum()
tmp = tmp / tmp.sum()
sns.heatmap(tmp.T, annot=True, fmt='0.2f');
```

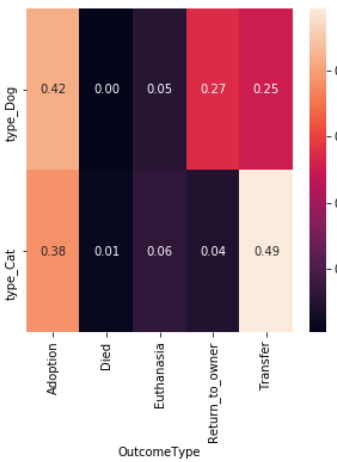
데이터 시각화 파트이다.
성별과 애완동물의 종류, 품종, 색깔과
입양된 수, 죽은 수, 안락사 수, 주인에게 돌아간
수, 이관된 수를 연관짓는 표들을 히트맵으로
표현했다.

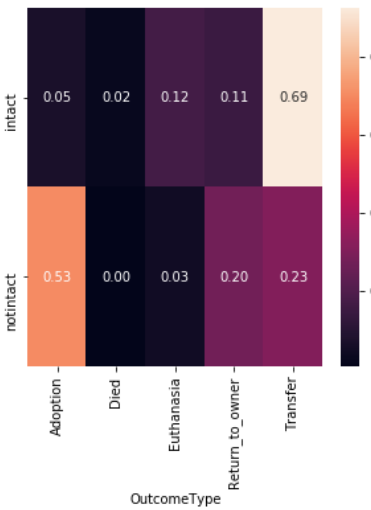

breed1_Abyssinian
breed1_Affenpinscher
breed1_Afghan Hound
breed1_Airedale Terrier
breed1_Akita
breed1_Alaskan Husky
breed1_Alaskan Malamute
breed1_American Bulldog
breed1_American Eskimo
breed1_American Foxhound
breed1_American Pit Bull Terrier
breed1_American Shorthair
breed1_American Staffordshire Terrier
breed1_Anatol Shepherd
breed1_Angora
breed1_Australian Cattle Dog
breed1_Australian Kelpie
breed1_Australian Shepherd
breed1_Australian Terrier
breed1_Balinese
breed1_Basenji
breed1_American Bulldog
breed1_American Eskimo
breed1_American Foxhound
breed1_American Pit Bull Terrier
breed1_American Shorthair
breed1_American Staffordshire Terrier
breed1_Anatol Shepherd
breed1_Angora
breed1_Australian Cattle Dog
breed1_Australian Kelpie
breed1_Australian Shepherd
breed1_Australian Terrier
breed1_Balinese
breed1_Basenji
breed1_Basset Hound
breed1_Beagle
breed1_Bearded Collie
breed1_Beauceron
breed1_Bedlington Terr
breed1_Belgian Malinois
breed1_Belgian Sheepdog
breed1_Belgian Tervuren
breed1_Bengal
breed1_Bernese Mountain Dog
breed1_Bichon Frise
breed1_Black
breed1_Black Mouth Cur
breed1_Bloodhound
breed1_Blue Lacy
breed1_Bluetick Hound
breed1_Boerboel
breed1_Bombay
breed1_Border Collie
breed1_Border Terrier
breed1_Borzoi
breed1_Boston Terrier
breed1_Boxer
breed1_Boykin Span
breed1_British Shorthair
breed1_Brittany
breed1_Bruss Griffon
breed1_Bull Terrier
breed1_Bull Terrier Miniature
breed1_Bulldog
breed1_Bullmastiff
breed1_Burmese
breed1_Cairn Terrier
breed1_Canaan Dog
breed1_Cane Corso
breed1_Cardigan Welsh Corgi
breed1_Carolina Dog
breed1_Catahoula
breed1_Cavalier Span
breed1_Chesa Bay Retr
breed1_Chihuahua Longhair
breed1_Chihuahua Shorthair
breed1_Chihuahua Shorthair
breed1_Chinese Crested
breed1_Chinese Sharpei
breed1_Chow Chow
breed1_Cocker Spaniel
breed1_Collie Rough
breed1_Collie Smooth
breed1_Cornish Rex
breed1_Cymric
breed1_Dachshund
breed1_Dachshund Longhair
breed1_Dachshund Wirehair
breed1_Dalmatian
breed1_Devon Rex
breed1_Doberman Pinsch
breed1_Dogo Argentino
breed1_Dogue De Bordeaux
breed1_Domestic Longhair
breed1_Domestic Medium Hair
breed1_Domestic Shorthair
breed1_Dutch Shepherd
breed1_English Bulldog
breed1_English Cocker Spaniel
breed1_English Coonhound
breed1_English Foxhound
breed1_English Pointer
breed1_English Setter
breed1_English Shepherd
breed1_English Springer Spaniel
breed1_Entlebucher
breed1_Exotic Shorthair
breed1_Feist
breed1_Field Spaniel
breed1_Finnish Spitz
breed1_Flat Coat Retriever
breed1_French Bulldog
breed1_German Pinscher
breed1_German Shepherd
breed1_German Shorthair Pointer
breed1_German Wirehaired Pointer
breed1_Glen Of Imaal
breed1_Golden Retriever
breed1_Great Dane
breed1_Great Pyrenees
breed1_Greater Swiss Mountain Dog
breed1_Greyhound
breed1_Harrier
breed1_Havana Brown
breed1_Havanese
breed1_Himalayan
breed1_Hovawart
breed1_Ibizan Hound
breed1_Irish Setter
breed1_Irish Terrier
breed1_Irish Wolfhound
breed1_Italian Greyhound


breed1_Italian Greyhound
breed1_Jack Russell Terrier
breed1_Japanese Bobtail
breed1_Japanese Chin
breed1_Javanese
breed1_Jindo
breed1_Keeshond
breed1_Kuvasz
breed1_Labrador Retriever
breed1_Landseer
breed1_Leonberger
breed1_Lhasa Apso
breed1_Lowchen
breed1_Maine Coon
breed1_Maltese
breed1_Manchester Terrier
breed1_Mara
breed1_Mastiff
breed1_Mexican Hairless
breed1_Miniature Pinscher
breed1_Miniature Poodle
breed1_Miniature Schnauzer
breed1_Munchkin Longhair
breed1_Neapolitan Mastiff
breed1_Newfoundland
breed1_Norfolk Terrier
breed1_Norwegian Elkhound
breed1_Norwegian Forest Cat
breed1_Norwich Terrier
breed1_Nova Scotia Duck Tolling Retriever
breed1_Ocicat
breed1_Old English Bulldog
breed1_Old English Sheepdog
breed1_Otterhound
breed1_Papillon
breed1_Parson Russell Terrier
breed1_Patterdale Terr
breed1_Pbgv
breed1_Pekingese
breed1_Pembroke Welsh Corgi
breed1_Persian
breed1_Pharaoh Hound
breed1_Picardy Sheepdog
breed1_Pit Bull
breed1_Pixiebob Shorthair
breed1_Plott Hound
breed1_Podengo Pequeno
breed1_Pointer
breed1_Pomeranian
breed1_Port Water Dog
breed1_Presa Canario
breed1_Pug
breed1_Queensland Heeler
breed1_Ragdoll
breed1_Rat Terrier
breed1_Redbone Hound
breed1_Rottweiler
breed1_Russian Blue
breed1_Saluki
breed1_Samoyed
breed1_Schipperke
breed1_Schnauzer Giant
breed1_Scottish Terrier
breed1_Sealyham Terr
breed1_Shetland Sheepdog
breed1_Shiba Inu
breed1_Shih Tzu
breed1_Siamese
breed1_Siberian Husky
breed1_Silky Terrier
breed1_Skye Terrier
breed1_Smooth Fox Terrier
breed1_Snowshoe
breed1_Soft Coated Wheaten Terrier
breed1_Spanish Mastiff
breed1_Spanish Water Dog
breed1_Sphynx
breed1_Spinone Italiano
breed1_St. Bernard Rough Coat
breed1_St. Bernard Smooth Coat
breed1_Staffordshire
breed1_Standard Poodle
breed1_Standard Schnauzer
breed1_Swedish Vallhund
breed1_Swiss Hound
breed1_Tan Hound
breed1_Tibetan Spaniel
breed1_Tibetan Terrier
breed1_Tonkinese
breed1_Toy Fox Terrier
breed1_Toy Poodle
breed1_Treeing Cur
breed1_Treeing Tennesse Brindle
breed1_Treeing Walker Coonhound
breed1_Turkish Van
breed1_Unknown
breed1_Vizsla
breed1_Weimaraner
breed1_Welsh Springer Spanier
breed1_Welsh Terrier
breed1_West Highland
breed1_Whippet
breed1_Wire Hair Fox Terrier
breed1_Wirehaired Pointing Griffon
breed1_Yorkshire
breed1_Yorkshire Terrier
Adoption
Died
Euthanasia
Return_to_owner
Transfer


color1_Agouti
color1_Apricot
color1_Black
color1_Black Brindle
color1_Black Smoke
color1_Black Tabby
color1_Black Tiger
color1_Blue
color1_Blue Cream
color1_Blue Merle
color1_Blue Point
color1_Blue Smoke
color1_Blue Tabby
color1_Blue Tick
color1_Blue Tiger
color1_Brown
color1_Brown Brindle
color1_Brown Merle
color1_Brown Tabby
color1_Brown Tiger
color1_Buff
color1_Calico
color1_Calico Point
color1_Chocolate
color1_Chocolate Point
color1_Cream
color1_Cream Tabby
color1_Fawn
color1_Flame Point
color1_Gold
color1_Gray
color1_Gray Tabby
color1_Lilac Point
color1_Liver
color1_Liver Tick
color1_Lynx Point
color1_Orange
color1_Orange Tabby
color1_Orange Tiger
color1_Pink
color1_Red
color1_Red Merle
color1_Red Tick
color1_Ruddy
color1_Sable
color1_Seal Point
color1_Silver
color1_Silver Lynx Point
color1_Silver Tabby
color1_Tan
color1_Torbie
color1_Tortie
color1_Tortie Point
color1_Tricolor
color1_White
color1_Yellow
color1_Yellow Brindle
Adoption
Died
Euthanasia
Return_to_owner
Transfer


male
female
0.37
0.01
0.06
0.19
0.36
0.44
0.01
0.05
0.17
0.34
Adoption
Died
Euthanasia
Return_to_owner
Transfer
OutcomeType


type_Dog
type_Cat
0.42
0.00
0.05
0.27
0.25
0.38
0.01
0.06
0.04
0.49
Adoption
Died
Euthanasia
Return_to_owner
Transfer
OutcomeType


intact
notintact
0.05
0.02
0.12
0.11
0.69
0.53
0.00
0.03
0.20
0.23
Adoption
Died
Euthanasia
Return_to_owner
Transfer
OutcomeType

코드리뷰

```
labelsEncoded = pd.DataFrame(labelEncoder.transform(labels.values.flatten()), columns=['OutcomeType'])
featuresExtended = pd.concat([animalTypeDf, datetimeDf, sexuponOutcomeDf, ageuponOutcomeDf, color1Df, color2Df, breed1Df, breed2Df],
axis=1)
featureScaler = sklearn.preprocessing.StandardScaler()
featuresProcessed = featureScaler.fit_transform(featuresExtended)
```

```
featuresTrain, featuresValidate, labelsTrain, labelsValidate = sklearn.model_selection.train_test_split(featuresProcessed, labelsEnco
ded.values, test_size=0.2)
```

```
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'x': featuresTrain},
    y=labelsTrain,
    batch_size=128,
    num_epochs=50,
    shuffle=True)

validate_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'x': featuresValidate},
    y=labelsValidate,
    shuffle=False)
```

데이터들을 분석 모듈에 맞게
변환 하는 코드이다.

코드리뷰

```
def model_fn(features, labels, mode, params):
    layer = features['x']
    layer = tf.layers.dense(inputs=layer, units=1024, activation=tf.nn.relu)
    layer = tf.layers.dense(inputs=layer, units=512, activation=tf.nn.relu)
    layer = tf.layers.dense(inputs=layer, units=256, activation=tf.nn.relu)
    layer = tf.layers.dense(inputs=layer, units=128, activation=tf.nn.relu)
    if mode == tf.estimator.ModeKeys.TRAIN:
        layer = tf.layers.dropout(inputs=layer, rate=0.4, training=mode == tf.estimator.ModeKeys.TRAIN)
    # Logits Layer
    logits = tf.layers.dense(inputs=layer, units=params['num_classes'])

    predictions = {
        "classes": tf.argmax(input=logits, axis=1),
        "probabilities": tf.nn.softmax(logits, name="softmax_tensor")
    }

    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)

    weights = tf.gather(params['weights'], labels)
    # Calculate Loss (for both TRAIN and EVAL modes)
    loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits, weights=weights)

    # Configure the Training Op (for TRAIN mode)
    if mode == tf.estimator.ModeKeys.TRAIN:
        optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
        train_op = optimizer.minimize(
          loss=loss,
          global_step=tf.train.get_global_step())
        return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)
```

```
    # Add evaluation metrics (for EVAL mode)
    eval_metric_ops = {
        "accuracy": tf.metrics.accuracy(
            labels=labels, predictions=predictions["classes"]),
        "recall": tf.metrics.recall(
            labels=labels, predictions=predictions["classes"]),
        "precision": tf.metrics.precision(
            labels=labels, predictions=predictions["classes"])}
    return tf.estimator.EstimatorSpec(
        mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)
```

로지스틱 회귀모듈이다.

코드리뷰

```
classifier = tf.estimator.Estimator(
    model_fn=model_fn,
    params={'num_classes': 5,
            'weights': [1., 1., 1., 1., 1.]})
classifier.train(input_fn=train_input_fn)
print(classifier.evaluate(input_fn=validate_input_fn))
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /tmp/tmp7meg_wwz
INFO:tensorflow:Using config: {'_model_dir': '/tmp/tmp7meg_wwz', '_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_config': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_service': None, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7f93ce4bd400>, '_task_type': 'worker', '_task_id': 0, '_global_id_in_cluster': 0, '_master': '', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 1 into /tmp/tmp7meg_wwz/model.ckpt.
INFO:tensorflow:loss = 1.8745639324188232, step = 1
INFO:tensorflow:global_step/sec: 29.3311
INFO:tensorflow:loss = 1.09073007106781, step = 101 (3.411 sec)
INFO:tensorflow:global_step/sec: 30.2054
INFO:tensorflow:loss = 1.1195309162139893, step = 201 (3.311 sec)
INFO:tensorflow:global_step/sec: 29.7938
INFO:tensorflow:loss = 1.1301507949829102, step = 301 (3.356 sec)
INFO:tensorflow:global_step/sec: 30.3931
INFO:tensorflow:loss = 1.0926654338836667, step = 401 (3.292 sec)
INFO:tensorflow:global_step/sec: 28.0531
INFO:tensorflow:loss = 0.9053487777709961, step = 501 (3.563 sec)
INFO:tensorflow:global_step/sec: 29.5684
INFO:tensorflow:loss = 0.937595009803772, step = 601 (3.382 sec)
INFO:tensorflow:global_step/sec: 30.3007
INFO:tensorflow:loss = 0.9473569393157959, step = 701 (3.300 sec)
INFO:tensorflow:global_step/sec: 29.7379
INFO:tensorflow:loss = 1.1462548971176147, step = 801 (3.363 sec)
INFO:tensorflow:global_step/sec: 29.4612
INFO:tensorflow:loss = 1.0104730129241943, step = 901 (3.394 sec)
INFO:tensorflow:global_step/sec: 29.5818
INFO:tensorflow:loss = 0.9176842570304871, step = 1001 (3.383 sec)
INFO:tensorflow:global_step/sec: 30.0406
INFO:tensorflow:loss = 0.9952024221420288, step = 1101 (3.327 sec)
INFO:tensorflow:global_step/sec: 28.593
INFO:tensorflow:loss = 1.058089017868042, step = 7201 (3.499 sec)
INFO:tensorflow:global_step/sec: 29.8791
INFO:tensorflow:loss = 1.0672786235809326, step = 7301 (3.346 sec)
INFO:tensorflow:global_step/sec: 29.2798
INFO:tensorflow:loss = 0.9447389841079712, step = 7401 (3.417 sec)
INFO:tensorflow:global_step/sec: 30.1641
INFO:tensorflow:loss = 1.0958080291748047, step = 7501 (3.315 sec)
INFO:tensorflow:global_step/sec: 30.1241
INFO:tensorflow:loss = 1.0410939455032349, step = 7601 (3.318 sec)
INFO:tensorflow:global_step/sec: 29.6296
INFO:tensorflow:loss = 1.0275297164916992, step = 7701 (3.375 sec)
INFO:tensorflow:global_step/sec: 30.1306
INFO:tensorflow:loss = 1.0369606018066406, step = 7801 (3.319 sec)
INFO:tensorflow:global_step/sec: 30.3522
INFO:tensorflow:loss = 1.0006251335144043, step = 7901 (3.295 sec)
INFO:tensorflow:global_step/sec: 29.0828
INFO:tensorflow:loss = 1.0212688446044922, step = 8001 (3.441 sec)
INFO:tensorflow:global_step/sec: 28.5229
INFO:tensorflow:loss = 0.9941677451133728, step = 8101 (3.503 sec)
INFO:tensorflow:global_step/sec: 30.0817
INFO:tensorflow:loss = 0.9115267992019653, step = 8201 (3.324 sec)
INFO:tensorflow:global_step/sec: 30.3705
INFO:tensorflow:loss = 0.9938536882400513, step = 8301 (3.293 sec)
INFO:tensorflow:Saving checkpoints for 8353 into /tmp/tmp7meg_wwz/model.ckpt.
INFO:tensorflow:Loss for final step: 0.9898297190666199.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2021-12-15-23:19:40
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmp7meg_wwz/model.ckpt-8353
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2021-12-15-23:19:41
INFO:tensorflow:Saving dict for global step 8353: accuracy = 0.55312383, global_step = 8353, loss = 1.0881411, precision = 0.89698994, recall = 0.41312385
{'accuracy': 0.55312383, 'loss': 1.0881411, 'precision': 0.89698994, 'recall': 0.41312385, 'global_step': 8353}
```


```
classifier = tf.estimator.Estimator(
    model_fn=model_fn,
    params={'num_classes': 5,
            'weights': [1., 1., 1., 1., 1.]})
classifier.train(input_fn=train_input_fn)
print(classifier.evaluate(input_fn=validate_input_fn))
```


추정치 분류 학습하는 과정이다.

추정치 학습 추정치를 출력한다.

코드리뷰

```
raw_predictions = list(classifier.predict(input_fn=validate_input_fn))
predicted_classes = list(map(lambda x: x['classes'], raw_predictions))
sns.heatmap(sklearn.metrics.confusion_matrix(labelsValidate, np.array(predicted_classes).reshape((-1,1))), annot=True);
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpsxkpd2yu/model.ckpt-8353
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
```

학습 완료되어 예측한 것을 히트맵 그래프로 시각화 했다.

코드리뷰

```
from sklearn.metrics import log_loss

predicted_probs = list(map(lambda x: x['probabilities'], raw_predictions))
print(log_loss(labelsValidate, predicted_probs) )
```

```
1.0894578044202357
```

예측 결과 값

최대값이 34.5이고 최저가 0이다.
0에 가까울 수록 좋은 모델이다.

코드리뷰

```
testDf = pd.read_csv('../input/test.csv.gz')
ids = testDf[['ID']].copy()
featuresTest = testDf[['DateTime', 'AnimalType', 'SexuponOutcome', 'AgeuponOutcome', 'Breed', 'Color']]

def preprocessFeatures(featuresTest):
    datetimeDf = featuresTest.apply(mapDateTime, axis=1)
    sexuponOutcomeDf = featuresTest.apply(mapSexuponOutcome, axis=1)
    ageuponOutcomeDf = featuresTest.apply(mapAgeuponOutcome, axis=1)
    breedLabelDf = featuresTest.apply(mapBreed, axis=1)
    colorLabelDf = featuresTest.apply(mapColor, axis=1)

    color1Df = pd.DataFrame(colorEncoder.transform(colorLabelDf['color1']))
    color1Df.columns = colorEncoder.classes_
    color1Df = color1Df.add_prefix('color1_')
    color2Df = pd.DataFrame(colorEncoder.transform(colorLabelDf['color2']))
    color2Df.columns = colorEncoder.classes_
    color2Df = color2Df.add_prefix('color2_')
    breed1Df = pd.DataFrame(breedEncoder.transform(breedLabelDf['breed1']))
    breed1Df.columns = breedEncoder.classes_
    breed1Df = breed1Df.add_prefix('breed1_')
    breed2Df = pd.DataFrame(breedEncoder.transform(breedLabelDf['breed2']))
    breed2Df.columns = breedEncoder.classes_
    breed2Df = breed2Df.add_prefix('breed2_')
    animalTypeDf = pd.DataFrame(animalTypeEncoder.transform(featuresTest['AnimalType']))
    animalTypeDf.columns = animalTypeEncoder.classes_[[1]]
    animalTypeDf = animalTypeDf.add_prefix('type_')

    featuresExtended = pd.concat([animalTypeDf, datetimeDf, sexuponOutcomeDf, ageuponOutcomeDf, color1Df, color2Df, breed1Df, breed2Df], axis=1)
    featuresProcessed = featureScaler.transform(featuresExtended)
    return featuresProcessed

featuresProcessed = preprocessFeatures(featuresTest)

test_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'x': featuresProcessed},
    shuffle=False)

raw_predictions = list(classifier.predict(input_fn=test_input_fn))
predicted_probs = list(map(lambda x: x['probabilities'], raw_predictions))
pd.concat([ids, pd.DataFrame(predicted_probs, columns=labelEncoder.inverse_transform(np.arange(0, 5)))], axis=1).to_csv('submission.csv', index=False)
```

위 코드들을 간략하게 필요한 코드만 사용하여 나온 히트맵과 마지막 결과값을 CSV파일로 변환 하는 코드이다.

- Q&A