Entrée [22]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import tensorflow_docs as tfdocs
import tensorflow_docs.plots
import tensorflow_docs.modeling
from sklearn.preprocessing import LabelEncoder
from math import *
from sklearn.preprocessing import OneHotEncoder
from tqdm.auto import tqdm
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn import preprocessing
```

Entrée [23]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Entrée [24]:
```
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))
mirrored_strategy = tf.distribute.MirroredStrategy()
```

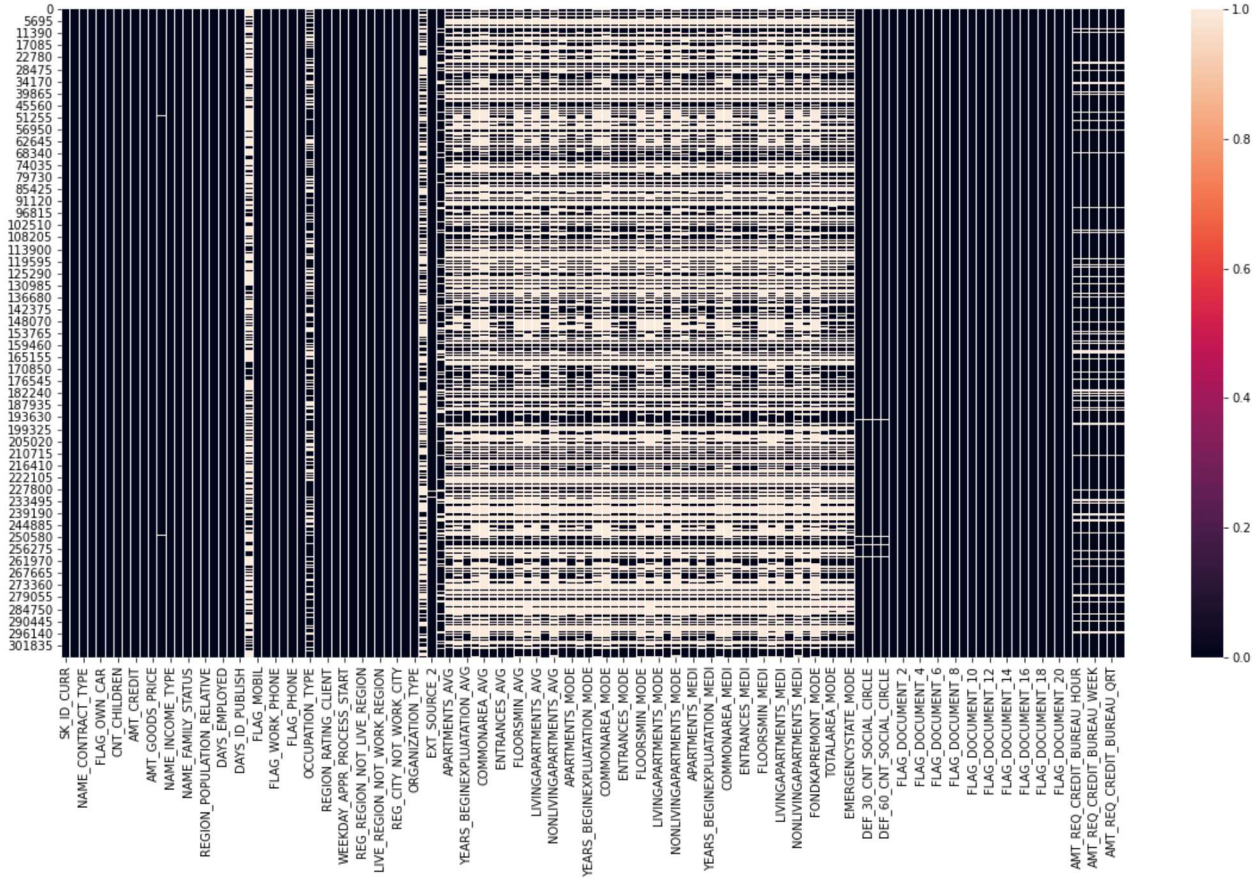
```
Num GPUs Available:  1
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:GPU:0',)
```

Entrée [25]:
```
df = pd.read_csv('home-credit-default-risk//application_train.csv')
```

visualisation data

Entrée [5]:
```
plt.figure(figsize=(20,10))
sns.heatmap(df.isna())
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x160736cc048>
```



valeur vide ou null

Entrée [6]:
```
(df.isna().sum()/df.shape[0]).sort_values()
```

```
CNT_CHILDREN                 0.000000
NAME_INCOME_TYPE             0.000000
NAME_EDUCATION_TYPE          0.000000
NAME_FAMILY_STATUS           0.000000
REGION_RATING_CLIENT         0.000000
REGION_POPULATION_RELATIVE   0.000000
DAYS_BIRTH                   0.000000
DAYS_EMPLOYED                0.000000
DAYS_REGISTRATION            0.000000
DAYS_ID_PUBLISH              0.000000
AMT_INCOME_TOTAL             0.000000
FLAG_OWN_REALTY              0.000000
CODE_GENDER                  0.000000
NAME_CONTRACT_TYPE           0.000000
FLAG_MOBIL                   0.000000
FLAG_EMP_PHONE               0.000000
FLAG_WORK_PHONE              0.000000
FLAG_CONT_MOBILE             0.000000
FLAG_PHONE                   0.000000
TARGET                       0.000000
```

Entrée [7]:
```
#on suppr toutes les colonnes à plus de 90% de val manquante
df = df[df.columns[(df.isna().sum()/df.shape[0]) < 0.9]]
```

Variables du csv

Entrée [8]:
```
df.dtypes.value_counts()
```

Out[8]:
```
float64    65
int64      41
object     16
dtype: int64
```

Entrée [ ]:
```
pbar = tqdm(total=1)
for col in df.select_dtypes('float'):
    plt.figure()
    sns.distplot(df[col])
    pbar.update(1/len(df.select_dtypes('float')))
pbar.close()
```

Entrée [ ]:
```
pbar = tqdm(total=1)
for col in df.select_dtypes('object'):
    print(f'{col :-<40} => {df[col].unique()}')
    plt.figure()
    df[col].value_counts(normalize=True, dropna=False).plot.pie()
    pbar.update(1/len(df.select_dtypes('object')))
pbar.close()
```

encoder des 16 colonnes objects

Entrée [26]:
```
le = []
for i,col in enumerate(df.select_dtypes('object').columns):
    try:
        le.append(LabelEncoder())
        df[col] = le[i].fit_transform(df[col].astype(str))
    except:
        print('error : '+col)
```

difference entre positif et negatif

Entrée [10]:
```
df.dtypes.value_counts()
```

Out[10]:
```
float64    65
int64      41
int32      16
dtype: int64
```

Entrée [11]:
```
positif = df[df['TARGET'] == 1 ]
negatif = df[df['TARGET'] != 1 ]
```

Entrée [12]:
```
df_int32_columns = df.select_dtypes('int32').columns
df_int64_columns = df.select_dtypes('int64').columns
df_float_columns = df.select_dtypes('float').columns
```

Entrée [ ]:
```
pbar = tqdm(total=1)
x=0
y=0
nbr = 3
fig, axs = plt.subplots(ceil(len(df_int32_columns)/nbr), nbr,  figsize=(20, 20))
for col in (df_int32_columns):
    if(x>=nbr):
        x=0
        y+=1
    sns.distplot(positif[col], label='positif', ax=axs[y,x])
    sns.distplot(negatif[col], label='negatif', ax=axs[y,x])
    axs[y,x].legend()
    x+=1
    pbar.update(1/len(df_int32_columns))
pbar.close()
```

Entrée [ ]:
```
pbar = tqdm(total=1)
x=0
y=0
nbr = 3
fig, axs = plt.subplots(ceil(len(df_int64_columns)/nbr), nbr,  figsize=(20, 80))
for col in (df_int64_columns):
    if(x>=nbr):
        x=0
        y+=1
    sns.distplot(positif[col], label='positif', ax=axs[y,x])
    sns.distplot(negatif[col], label='negatif', ax=axs[y,x])
    axs[y,x].legend()
    x+=1
    pbar.update(1/len(df_int64_columns))
pbar.close()
```

Entrée [ ]:
```
pbar = tqdm(total=1)
x=0
y=0
nbr = 3
fig, axs = plt.subplots(ceil(len(df_float_columns)/nbr), nbr,  figsize=(20, 120))
for col in (df_float_columns):
    if(x>=nbr):
        x=0
        y+=1
    sns.distplot(positif[col], label='positif', ax=axs[y,x])
    sns.distplot(negatif[col], label='negatif', ax=axs[y,x])
    axs[y,x].legend()
    x+=1
    pbar.update(1/len(df_float_columns))
pbar.close()
```

nouveau dataTrain

Entrée [27]:
```
data_train = df[['TARGET','EMERGENCYSTATE_MODE','HOUSETYPE_MODE','OCCUPATION_TYPE','WEEKDAY_APPR_PROCESS_START','ORGAN
data_train.head()
```

Out[27]:

	TARGET	EMERGENCYSTATE_MODE	HOUSETYPE_MODE	OCCUPATION_TYPE	WEEKDAY_APPR_PROCESS_START	ORGANIZATION_TYPE	COD
0	1	0	0	8	6	5	
1	0	0	0	3	1	39	
2	0	2	1	8	1	11	
3	0	2	1	8	6	5	
4	0	2	1	3	4	37	

Entrée [14]:
```
(data_train.isna().mean()).sort_values()
```

Out[14]:
```
TARGET                        0.000000
DAYS_REGISTRATION             0.000000
REGION_POPULATION_RELATIVE    0.000000
AMT_CREDIT                    0.000000
FLAG_DOCUMENT_3               0.000000
REG_CITY_NOT_WORK_CITY        0.000000
FLAG_PHONE                    0.000000
DAYS_BIRTH                    0.000000
FONDKAPREMONT_MODE            0.000000
NAME_EDUCATION_TYPE           0.000000
DAYS_ID_PUBLISH               0.000000
FLAG_OWN_REALTY               0.000000
EMERGENCYSTATE_MODE           0.000000
NAME_INCOME_TYPE              0.000000
OCCUPATION_TYPE               0.000000
WEEKDAY_APPR_PROCESS_START    0.000000
HOUSETYPE_MODE                0.000000
CODE_GENDER                   0.000000
FLAG_OWN_CAR                  0.000000
ORGANIZATION_TYPE             0.000000
AMT_GOODS_PRICE               0.000904
EXT_SOURCE_2                  0.002146
OBS_30_CNT_SOCIAL_CIRCLE      0.003320
OBS_60_CNT_SOCIAL_CIRCLE      0.003320
DEF_30_CNT_SOCIAL_CIRCLE      0.003320
DEF_60_CNT_SOCIAL_CIRCLE      0.003320
EXT_SOURCE_3                  0.198253
EXT_SOURCE_1                  0.563811
dtype: float64
```

Entrée [15]:
```
data_train['EMERGENCYSTATE_MODE'].isna().sum()
```

Out[15]:
```
0
```

Entrée [ ]:
```
pbar = tqdm(total=1)
for col in data_train.columns:
    if((data_train[col].isna().sum()) > 0):
        print(f'{col :-<40} => {data_train[col].unique()}')
        plt.figure()
        data_train[col].value_counts(normalize=True, dropna=False).plot.pie()
    pbar.update(1/len(data_train.columns))
pbar.close()
```

Valeurs vide

Entrée [28]:
```
columns = (data_train.isna().mean()).sort_values() >0
columns_index = columns.index[columns.values == True]
```

Entrée [29]:

```python
imp_median= SimpleImputer(missing_values=np.nan, strategy='median')

pbar = tqdm(total=1)
for column in columns_index:

    imp_median= imp_median.fit(data_train[[column]])
    data_train[column] = imp_median.transform(data_train[[column]]).ravel()

    pbar.update(1/(len(columns_index)))
pbar.close()
```

100% 1.0/1 [00:00<00:00, 3.10it/s]

```
<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()
<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()
<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()
<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()
<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()
<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()



<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()
<ipython-input-29-e140b110cf52>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  data_train[column] = imp_median.transform(data_train[[column]]).ravel()
```

Entrée [18]:
```
(data_train.isna().mean()).sort_values()
```

Out[18]:
```
TARGET                         0.0
DEF_30_CNT_SOCIAL_CIRCLE       0.0
OBS_60_CNT_SOCIAL_CIRCLE       0.0
EXT_SOURCE_3                   0.0
EXT_SOURCE_2                   0.0
EXT_SOURCE_1                   0.0
DAYS_REGISTRATION              0.0
REGION_POPULATION_RELATIVE     0.0
AMT_GOODS_PRICE                0.0
AMT_CREDIT                     0.0
FLAG_DOCUMENT_3                0.0
REG_CITY_NOT_WORK_CITY         0.0
FLAG_PHONE                     0.0
DAYS_ID_PUBLISH                0.0
DAYS_BIRTH                     0.0
FONDKAPREMONT_MODE             0.0
NAME_EDUCATION_TYPE            0.0
NAME_INCOME_TYPE               0.0
FLAG_OWN_REALTY                0.0
FLAG_OWN_CAR                   0.0
CODE_GENDER                    0.0
ORGANIZATION_TYPE              0.0
WEEKDAY_APPR_PROCESS_START     0.0
OCCUPATION_TYPE                0.0
HOUSETYPE_MODE                 0.0
EMERGENCYSTATE_MODE            0.0
OBS_30_CNT_SOCIAL_CIRCLE       0.0
DEF_60_CNT_SOCIAL_CIRCLE       0.0
dtype: float64
```

normalisation de la data

Entrée [19]:
```
data_train.head()
```

Out[19]:

	TARGET	EMERGENCYSTATE_MODE	HOUSETYPE_MODE	OCCUPATION_TYPE	WEEKDAY_APPR_PROCESS_START	ORGANIZATION_TYPE	COD
0	1	0	0	8	6	5	
1	0	0	0	3	1	39	
2	0	2	1	8	1	11	
3	0	2	1	8	6	5	
4	0	2	1	3	4	37	

Entrée [30]:
```
from sklearn.preprocessing import RobustScaler
x = data_train.values #returns a numpy array
rs = RobustScaler()
x_scaled = rs.fit_transform(x)
df = pd.DataFrame(x_scaled, columns=['TARGET','EMERGENCYSTATE_MODE','HOUSETYPE_MODE','OCCUPATION_TYPE','WEEKDAY_APPR_P
df.head()
```

Out[30]:

	TARGET	EMERGENCYSTATE_MODE	HOUSETYPE_MODE	OCCUPATION_TYPE	WEEKDAY_APPR_PROCESS_START	ORGANIZATION_TYPE	COD
0	1.0	0.0	-1.0	-0.166667	0.50	-0.666667	
1	0.0	0.0	-1.0	-0.583333	-0.75	0.142857	
2	0.0	1.0	0.0	-0.166667	-0.75	-0.523810	
3	0.0	1.0	0.0	-0.166667	0.50	-0.666667	
4	0.0	1.0	0.0	-0.583333	0.00	0.095238	

train et test

methode 3 SMOTE

Entrée [31]:
```
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
import imblearn
from collections import Counter
```

Entrée [32]:
```
X = data_train
y = data_train.TARGET

smote = SMOTE(random_state=0)
X_resampled, y_resampled = smote.fit_sample(X, y)

X_resampled = pd.DataFrame(X_resampled, columns=X.columns)
data_train = X_resampled
```

train test split

Entrée [33]:
```
X_train, X_test, y_train, y_test = train_test_split(data_train.drop(columns=['TARGET']), data_train['TARGET'], test_si
```

Reseaux neuronnes

Model

Entrée [42]:
```
model10 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(20, activation='relu', input_shape=(1,27)),
    #tf.keras.layers.Dropout(0.5),
    #tf.keras.layers.Dense(30, activation='relu',kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    #tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Entrée [43]:
```
model10.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 1, 20)             560
_________________________________________________________________
dense_7 (Dense)              (None, 1, 1)              21
=================================================================
Total params: 581
Trainable params: 581
Non-trainable params: 0
_________________________________________________________________
```

Entrée [44]:
```
model10.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

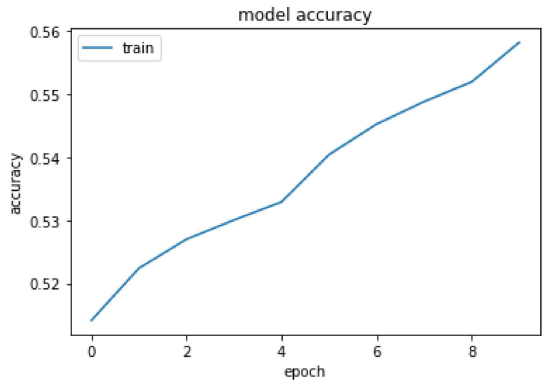
Entrainement

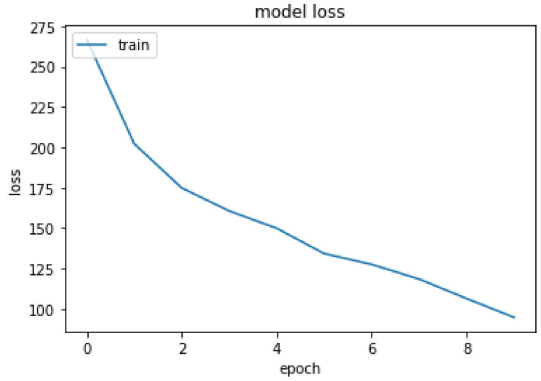
Entrée [45]:
```
history = model10.fit(X_train,
                      y_train,
                    epochs=10,
                    batch_size= 40,
                    callbacks=[tf.keras.callbacks.EarlyStopping(monitor='loss', patience=10),
                               tfdocs.modeling.EpochDots(),
                               tf.keras.callbacks.ModelCheckpoint("weights.best.hdf5",
                                                                  monitor='accuracy',
                                                                  verbose=0,
                                                                  save_best_only=True,
                                                                  save_weights_only=False,
                                                                  mode='max',
                                                                  periode=1)])
```

```
11308/11308 [==============================] - 26s 2ms/step - loss: 266.8862 - accuracy: 0.5141
Epoch 2/10
11308/11308 [==============================] - 25s 2ms/step - loss: 202.2263 - accuracy: 0.5224
Epoch 3/10
11308/11308 [==============================] - 24s 2ms/step - loss: 174.8956 - accuracy: 0.5270
Epoch 4/10
11308/11308 [==============================] - 24s 2ms/step - loss: 160.6586 - accuracy: 0.5300
Epoch 5/10
11308/11308 [==============================] - 24s 2ms/step - loss: 149.9458 - accuracy: 0.5329
Epoch 6/10
11308/11308 [==============================] - 23s 2ms/step - loss: 134.2422 - accuracy: 0.5404
Epoch 7/10
11308/11308 [==============================] - 24s 2ms/step - loss: 127.5847 - accuracy: 0.5453
Epoch 8/10
11308/11308 [==============================] - 24s 2ms/step - loss: 118.4931 - accuracy: 0.5488
Epoch 9/10
11308/11308 [==============================] - 25s 2ms/step - loss: 106.4252 - accuracy: 0.5520
Epoch 10/10
11308/11308 [==============================] - 24s 2ms/step - loss: 94.7649 - accuracy: 0.5582
```

Courbe apprentissage cnn

Entrée [46]:
```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Save load model

Entrée [48]:
```
model10.save('model10.h5')
```

Entrée [ ]:
```
model10 = tf.keras.models.load_model('model10.h5')
```

Evaluation du model

Entrée [49]:
```
test_loss, test_acc = model10.evaluate(X_test,y_test)
```

```
WARNING:tensorflow:Model was constructed with shape (None, 1, 27) for input Tensor("dense_6_input:0", shape=(None,
1, 27), dtype=float32), but it was called on an input with incompatible shape (None, 27).
3534/3534 [==============================] - 6s 2ms/step - loss: 95.2765 - accuracy: 0.5313
```

ensemble classifier

Entrée [50]:
```
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import *
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
```

Entrée [51]:
```
model1 = SGDClassifier(random_state=0)
model2 = DecisionTreeClassifier(max_features=7, random_state=0)
model3 = KNeighborsClassifier(n_neighbors=1, metric='manhattan', n_jobs=-1)
model4 = RandomForestClassifier(n_estimators=9, max_features=9,  n_jobs=-1)
model5 = AdaBoostClassifier()
model6 = GaussianNB()
```

Entrée [ ]:
```
param_grid = {'n_neighbors':np.arange(1,10),
              'metric':['euclidean','manhattan']}

grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
grid.best_score_
```

Entrée [ ]:
```
grid.best_params_
```

Entrée [ ]:
```
param_grid = {'n_estimators':np.arange(1,10),
              'max_features':np.arange(1,10)}

grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
grid.best_score_
```

Entrée [ ]:
```
grid.best_params_
```

Entrée [ ]:
```
param_grid = {'max_features':np.arange(1,10)}

grid = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
grid.best_score_
```

Entrée [ ]:
```
grid.best_params_
```

votingClassifier

Entrée [52]:
```
#voting hard voting, soft voting
model11 = VotingClassifier([#('sgd', model1),
                            ('tree', model2),
                            ('kn', model3),
                            ('rdforest',model4),
                            ('ada',model5),
                            #('bayes',model6)
],
                           voting='soft')
```

Entrée [58]:
```
for model in (model1, model2, model3, model4, model5, model6, model11):
    model.fit(X_train, y_train)
    print(model.__class__.__name__, model.score(X_test, y_test))
    print(confusion_matrix(y_test, model.predict(X_test)))
```

```
SGDClassifier 0.5000309529073623
[[    2 56534]
 [    0 56539]]
DecisionTreeClassifier 0.863046650453239
[[48038  8498]
 [ 6988 49551]]
KNeighborsClassifier 0.8533009064780013
[[44623 11913]
 [ 4675 51864]]
RandomForestClassifier 0.922210921954455
[[53645  2891]
 [ 5905 50634]]
AdaBoostClassifier 0.8713154985629007
[[49209  7327]
 [ 7224 49315]]
GaussianNB 0.5826752155648905
[[23630 32906]
 [14283 42256]]
VotingClassifier 0.9310811408357285
[[53261  3275]
 [ 4518 52021]]
```

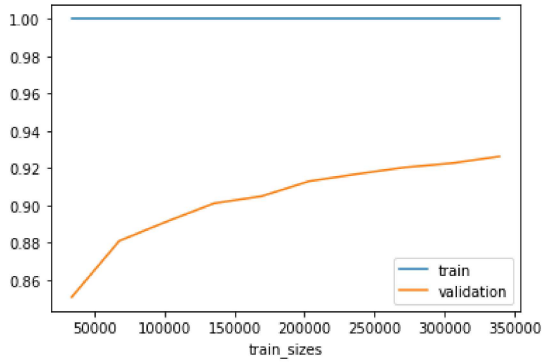
courbe d aprentissage

Entrée [59]:
```
from sklearn.model_selection import learning_curve
```

Entrée [60]:
```
N, train_score, val_score = learning_curve(model11, X_train, y_train, train_sizes=np.linspace(0.1,1.0,10), cv=4)

plt.plot(N, train_score.mean(axis=1), label='train')
plt.plot(N, val_score.mean(axis=1), label='validation')
plt.xlabel('train_sizes')
plt.legend()
```

Out[60]:
```
<matplotlib.legend.Legend at 0x1c170df2bb0>
```



Entrée [ ]:
```
```