mohsensho / deepLearningWithTensorflowKeras Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

deepLearningWithTensorflowKeras / houseloan-data-analysis.ipynb

mohsensho fixing data path

History

0 contributors

```
1 lines (1 sloc) | 30.6 KB
```

In [ ]:

```
import pandas as pd
import sklearn
import numpy as np
import matplotlib.pyplot as plt
import os
import warnings
import seaborn as sns
from sklearn.preprocessing import OneHotEncod
from sklearn.datasets import make_blobs
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardSca
from sklearn.svm import LinearSVC
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import LogisticRegr
from sklearn.metrics import roc_auc_score
from sklearn.calibration import CalibratedCla
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClas
from sklearn.metrics import accuracy_score
from sklearn.linear_model import SGDClassifie
import plotly.offline as py
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode
from sklearn.model_selection import train_tes
init_notebook_mode(connected=True)
import cufflinks as cf
cf.go_offline()
import pickle
import gc
import lightgbm as lgb
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [ ]:

```
house_loan=pd.read_csv('loan_data_.csv')
house_loan.describe()
```

In [ ]:

```
house_loan.columns
```

In [ ]:

```
house_loan.info()
```

In [ ]:

```
house_loan.isnull().sum()
```

In [ ]:

```
house_loan.head()
```

In [ ]:

```
defaulters=(house_loan.TARGET==1).sum()
payers=(house_loan.TARGET==0).sum()
print((defaulters/payers)*100)
```

In [ ]:

```
without_id=[column for column in house_loan.c

#check for duplicate values
na=house_loan[house_loan.duplicated(subset=wi
print("Duplicates are: ",na.shape[0])
```

In [ ]:

```
house_loan.TARGET.value_counts().plot(kind='p
```

In [ ]:

```
import matplotlib as plt
```

In [ ]:

```
shuffled_data=house_loan.sample(frac=1,random
unpaid_home_loan=shuffled_data.loc[shuffled_d
paid_home_loan=shuffled_data.loc[shuffled_dat
normalised_home_loan=pd.concat([unpaid_home_l
normalised_home_loan.TARGET.value_counts().pl
```

In [ ]:

```
import tensorflow as tf
```

In [ ]:

```
normalised_home_loan.info()
```

In [ ]:

```
normalised_home_loan.head
```

In [ ]:

```
normalised_home_loan.dropna(axis=0)
normalised_home_loan.info()
```

In [ ]:

```
normalised_home_loan.isnull().sum()
```

In [ ]:

```
#print(normalised_home_loan.apply())
```

In [ ]:

```
print(pd.unique(normalised_home_loan.AMT_REQ_
print(pd.unique(normalised_home_loan.AMT_REQ_
print(pd.unique(normalised_home_loan.AMT_REQ_
print(pd.unique(normalised_home_loan.AMT_REQ_
print(pd.unique(normalised_home_loan.AMT_REQ_
```

In [ ]:

```
normalised_home_loan.dropna(axis=0)
```

In [ ]:

```
print(normalised_home_loan.info())
print(normalised_home_loan.isnull().sum())
```

In [ ]:

```
normalised_home_loan.TARGET.value_counts().pl
```

In [ ]:

```
normalised_home_loan.NAME_CONTRACT_TYPE.value
#high amount of cash loans
```

In [ ]:

```
normalised_home_loan.CODE_GENDER.value_counts
#roughly equal amount
```

In [ ]:

```
normalised_home_loan.FLAG_OWN_CAR.value_count
```

In [ ]:

```
normalised_home_loan.CNT_CHILDREN.value_count
```

In [ ]:

```
#!pip install chart_studio


cf.set_config_file(theme='polar')

normalised_home_loan[normalised_home_loan['AM
   xTitle = 'Total Income', yTitle ='Count of
             title='Distribution of AMT_INCOM
```

In [ ]:

```
(normalised_home_loan[normalised_home_loan['A
```

In [ ]:

```
#print((normalised_home_loan[normalised_home_
print((normalised_home_loan[normalised_home_l
print((normalised_home_loan[normalised_home_l
#as number of children is increasing lone def
```

In [ ]:

```
print((normalised_home_loan[normalised_home_l
print((normalised_home_loan[normalised_home_l

#people with own cars are slighlty more likel
```

In [ ]:

```
print((normalised_home_loan[normalised_home_l
print((normalised_home_loan[normalised_home_l

#men more likely to default in payment of loa
```

In [ ]:

```
print((normalised_home_loan[normalised_home_l
print((normalised_home_loan[normalised_home_l

#cash loans have a higher percent of defaulte
```

In [ ]:

```
normalised_home_loan=normalised_home_loan.sam
```

In [ ]:

```
from sklearn.preprocessing import OrdinalEnco

ordenc=OrdinalEncoder()
normalised_home_loan['NAME_CONTRACT_TYPE_CODE
print(normalised_home_loan[['NAME_CONTRACT_TY
print(normalised_home_loan['NAME_CONTRACT_TYP
```

In [ ]:

```
normalised_home_loan['CODE_GENDER_CODE']=orde
print(normalised_home_loan[['CODE_GENDER','CO
print(normalised_home_loan['CODE_GENDER_CODE'
```

In [ ]:

```
#2 other values in code_gender
normalised_home_loan.loc[normalised_home_loan
```

In [ ]:

```
normalised_home_loan['FLAG_OWN_CAR_CODE']=ord
print(normalised_home_loan[['FLAG_OWN_CAR','F
print(normalised_home_loan['FLAG_OWN_CAR_CODE
```

In [ ]:

```
normalised_home_loan['CNT_CHILDREN_CODE']=ord
print(normalised_home_loan[['CNT_CHILDREN_COD
print(normalised_home_loan['CNT_CHILDREN_CODE
```

In [ ]:

```
normalised_home_loan=normalised_home_loan.sam
```

In [ ]:

```
normalised_home_loan['TARGET'].value_counts()
```

In [ ]:

```
y=normalised_home_loan.TARGET
```

In [ ]:

```
#y=y.sample(frac=1,random_state=45)
```

In [ ]:

```
normalised_home_loan_features=['SK_ID_CURR','
```

In [ ]:

```
from sklearn.model_selection import train_tes
```

In [ ]:

```
X=normalised_home_loan[normalised_home_loan_f
```

In [ ]:

```
#X=X.sample(frac=1,random_state=45)
```

In [ ]:

```
blobs_random_seed = 42
centers = [(0,0), (5,5)]
cluster_std = 1
frac_test_split = 0.33
num_features_for_samples = 2
num_samples_total = 49650

# Generate data
inputs, targets = make_blobs(n_samples = num_

X_train,X_test,y_train,y_test=train_test_spli
```

In [ ]:

```
print(X_train.shape, X_test.shape, y_train.sh
```

In [ ]:

```
plt.pyplot.scatter(X_train[:,0], X_train[:,1]
plt.pyplot.title('Linearly separable data')
plt.pyplot.xlabel('X1')
plt.pyplot.ylabel('X2')
plt.pyplot.show()
```

In [ ]:

```
from sklearn import svm
from sklearn.metrics import ConfusionMatrixDi
```

In [ ]:

```
clf=svm.SVC(kernel='linear')
```

In [ ]:

```
clf=clf.fit(X_train,y_train)
```

In [ ]:

```
predictions = clf.predict(X_test)

# Generate confusion matrix
matrix = ConfusionMatrixDisplay.from_predicti
plt.pyplot.title('Confusion matrix for our cl
plt.pyplot.show(matrix)
plt.pyplot.show()
```

In [ ]:

```
from sklearn.metrics import precision_score,
```

In [ ]:

```
print(precision_score(y_test, predictions))
print(recall_score(y_test, predictions))
print(f1_score(y_test,predictions,average=Nor
```

In [ ]:

```
support_vectors = clf.support_vectors_

# Visualize support vectors
plt.pyplot.scatter(X_train[:,0], X_train[:,1]
plt.pyplot.scatter(support_vectors[:,0], supp
plt.pyplot.title('Linearly separable data wit
plt.pyplot.xlabel('X1')
plt.pyplot.ylabel('X2')
plt.pyplot.show()
```

In [ ]:

```
from mlxtend.plotting import plot_decision_re
```

In [ ]:

```
plot_decision_regions(X_test, y_test, clf=clf
plt.pyplot.show()
```

In [ ]: