1.Extract file structure

for dirname, \_, filenames in os.walk('D:\\ML\\Kaggle\\SalesForecasting\\competitive-data-science-predict-future-sales\\'):

for filename in filenames:

print(os.path.join(dirname, filename))

2.Read csv file

items = pd.read\_csv('items.csv')

3.Filter data

train[train.item\_cnt\_day<0]

4.Visualize data

plt.figure(figsize=(10,4))

plt.xlim(-100, 3000)

flierprops = dict(marker='^', markerfacecolor='purple', ms=12,

linestyle='none', markeredgecolor='black')

sns.boxplot(x=train.item\_cnt\_day, flierprops=flierprops)

5.Filter Multiple columns

train = train[(train.item\_price < 300000 )& (train.item\_cnt\_day < 1000)]

6.Filter and reset index

train = train[train.item\_price > 0].reset\_index(drop = True)

7.Display filtered data as outputs

display(train[train.item\_cnt\_day<0])

8.Find duplicate data

display(shops[shops.shop\_name.duplicated()])

9.Sort data

display(train[train.shop\_id==0].sort\_values("date\_block\_num"))

10.Find unique values

train[train.shop\_id==2].date\_block\_num.unique()

11.Graph bivariate data using unique values and group by

shop\_id\_month\_list = []

for i in range(0,60):

unq = train[train.shop\_id==i].date\_block\_num.unique()

#print(f"{i} {train[train.shop\_id==i].date\_block\_num.unique()}")

for j in unq:

shop\_id\_month\_list.append([i, j])

plt.figure(figsize = (12,8))

x, y = zip(\*shop\_id\_month\_list)

sns.scatterplot(x,y, s=50)

plt.show()

12.Replace data

train.loc[train.shop\_id == 0, 'shop\_id'] = 57

13.Replace text data

shops.loc[ shops.shop\_name == 'Сергиев Посад ТЦ "7Я"',"shop\_name" ] = 'СергиевПосад ТЦ "7Я"'

13.Split and select lambda function used

shops["category"] = shops.shop\_name.str.split(" ").map( lambda x: x[1] )

14.Create others category for lower frequency

category = []

for cat in shops.category.unique():

if len(shops[shops.category == cat]) >= 5:

category.append(cat)

shops.category = shops.category.apply( lambda x: x if (x in category) else "other" )

15.Label encoder implementation

shops["shop\_category"] = LabelEncoder().fit\_transform( shops.category )

16.select subset of columns

shops = shops[["shop\_id", "shop\_category", "shop\_city"]]

17.display first 6 rows

display(shops.head())

18.Usage of lambda function

cats["type\_code"] = cats.item\_category\_name.apply( lambda x: x.split(" ")[0] ).astype(str)

19.replace text data multiple conditions

cats.loc[ (cats.type\_code == "Игровые")| (cats.type\_code == "Аксессуары"), "type\_code" ] = "Игры"

20.code data

category = []

for cat in cats.type\_code.unique():

if len(cats[cats.type\_code == cat]) >= 5:

category.append( cat )

cats.type\_code = cats.type\_code.apply(lambda x: x if (x in category) else "etc")

21.Using if condition in lambda

cats["subtype"] = cats.split.apply(lambda x: x[1].strip() if len(x) > 1 else x[0].strip())

22.Create new function for text cleansing

import re

def name\_correction(x):

x = x.lower() # all letters lower case

x = x.partition('[')[0] # partition by square brackets

x = x.partition('(')[0] # partition by curly brackets

x = re.sub('[^A-Za-z0-9А-Яа-я]+', ' ', x) # remove special characters

x = x.replace(' ', ' ') # replace double spaces with single spaces

x = x.strip() # remove leading and trailing white space

return x

Apply that function

name\_correction("PC, Цифровая версия]")

23.replace and lower

tmp = "AidfhoasgDFhs oghas Su!!["

re.sub("[^A-Z]+", " ", tmp).lower()

24.create function and apply

tmp = lambda x: x[:-1] if x !="0" else "0"

tmp("pc цифровая версия")

25.# replace special characters and turn to lower case

items["name2"] = items.name2.str.replace('[^A-Za-z0-9А-Яа-я]+', " ", regex=True).str.lower() # 括弧とかが削除されている

items["name3"] = items.name3.str.replace('[^A-Za-z0-9А-Яа-я]+', " ", regex=True).str.lower()

26.items = items.fillna('0')

Fill nulls with 0

Fill blanks with 0

27.encoind of data

'pс'.encode()

28.aggregate data

items.groupby(["type"]).agg({"item\_id": "count"}).reset\_index()

29.tmp=items.groupby(["type"]).agg({"item\_id": "count"})

tmp = tmp.reset\_index()

tmp.loc[(tmp.type == "0"), "item\_id"].values[0]

Aggregate data

30.create other after grouping

group\_sum = items.groupby(["type"]).agg({"item\_id": "count"})

group\_sum = group\_sum.reset\_index()

display(items[items.name2=="other"])

for cat in group\_sum.type.unique():

if group\_sum.loc[(group\_sum.type == cat), "item\_id"].values[0] <40:

# drop\_cols.append(cat)

items.loc[(items["type"] == cat), "name2"] = "other"

items.loc[(items["type"] == cat), "type"] = "other"

31.drop columns

items.drop(["item\_name", "name1"],axis = 1, inplace= True)

dataset.drop(labels = ['shop\_id','ID','item\_id'],inplace = True,axis = 1)

dataset.drop(labels = ['shop\_id','ID','item\_id'],inplace = True,axis = 1)

32..append using matrix

ts = time.time()

matrix = []

cols = ["date\_block\_num", "shop\_id", "item\_id"]

for i in range(34):

sales = train[train.date\_block\_num == i]

matrix.append( np.array(list( product( [i], sales.shop\_id.unique(), sales.item\_id.unique() ) ), dtype = np.int16) )

matrix = pd.DataFrame( np.vstack(matrix), columns = cols )

matrix["date\_block\_num"] = matrix["date\_block\_num"].astype(np.int8)

matrix["shop\_id"] = matrix["shop\_id"].astype(np.int8)

matrix["item\_id"] = matrix["item\_id"].astype(np.int16)

matrix.sort\_values( cols, inplace = True )

time.time()- ts

33.ts = time.time()

group = train.groupby( ["date\_block\_num", "shop\_id", "item\_id"] ).agg( {"item\_cnt\_day": ["sum"]} )

Group by sum

34.merge data

matrix = pd.merge( matrix, group, on = cols, how = "left" )

dataset = pd.merge(test, dataset, on = ['shop\_id','item\_id'], how = 'left')

35.matrix["item\_cnt\_month"] = matrix["item\_cnt\_month"].fillna(0).astype(np.float16)

Fill na values

36.convert to np.int8,16

test["date\_block\_num"] = test["date\_block\_num"].astype(np.int8)

test["shop\_id"] = test.shop\_id.astype(np.int8)

test["item\_id"] = test.item\_id.astype(np.int16)

37.rbind or concat columns

matrix = pd.concat([matrix, test.drop(["ID"],axis = 1)], ignore\_index=True, sort=False, keys=cols)

38.print data

print(f"shops:{shops.columns}")

39.aggregate and use group by

group = matrix.groupby( ["date\_block\_num"] ).agg({"item\_cnt\_month" : ["mean"]})

40.convert to float data type

matrix.date\_avg\_item\_cnt = matrix["date\_avg\_item\_cnt"].astype(np.float16)

41.check stored memory

del data

gc.collect()

42.xgboost model

ts = time.time()

dtrain = xgb.DMatrix(train\_X, train\_y)

dvalid = xgb.DMatrix(val\_X, val\_y)

params = {

"objective" : "reg:squarederror",

"eval\_metric" : "rmse"

}

results\_dict = {}

model = xgb.train(

params = params,

dtrain = dtrain,

evals = [(dtrain, "train"), (dvalid, "valid")],

num\_boost\_round = 1000,

early\_stopping\_rounds = 20,

evals\_result = results\_dict

)

time.time() - ts

43.Predict test data using xgboost

pred\_y = model.predict(xgb.DMatrix(val\_X)).clip(0, 20)

test\_y = model.predict(xgb.DMatrix(test\_X)).clip(0, 20)

submission = pd.DataFrame({

"ID": test.index,

"item\_cnt\_month": test\_y

})

submission.to\_csv('xgb\_submission.csv', index=False)

44.Compare train and test models

plt.plot(results\_dict["train"]["rmse"], color = "red", label = "train")

plt.plot(results\_dict["valid"]["rmse"], color = "blue", label = "valid")

plt.legend()

plt.show()

45.Variable importance

xgb.plot\_importance(model, importance\_type = "gain")

plt.show()

46.Graphviz configuration

Install graph viz from the website

Run this code to add path

import os os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'

Then run the graphviz code

import os

os.environ["PATH"] += os.pathsep + 'C:/Program Files/Graphviz/bin/'

fig = plt.figure(figsize=(50,30))

ax = plt.subplot()

xgb.plot\_tree(model, num\_trees=2, ax=ax, rankdir='LR')

# fig.savefig("img.png")

plt.show()

47.date time handling

from datetime import datetime

def timestamp(x):

return datetime.strptime(x, "%d.%m.%Y")

sales\_train['date'] = sales\_train['date'].apply(timestamp)

48.drop by index column

items.drop('item\_name',1,inplace = True)

items

49.Pivot table

dataset = sales\_train.pivot\_table(index = ['shop\_id','item\_id'],values = ['item\_cnt\_day'],columns = ['date\_block\_num'], fill\_value= 0)

dataset.reset\_index(inplace = True)

Dataset

sales\_data = sales\_data.pivot\_table(index = 'ID', columns='date\_block\_num', values = 'sum', aggfunc='sum')

50.Inplace explanation

When inplace=True is passed, the data is renamed in place (it returns nothing), so you'd use:

df.an\_operation(inplace=True)

When inplace=False is passed (this is the default value, so isn't necessary), performs the operation and returns a copy of the object, so you'd use:

df = df.an\_operation(inplace=False)

51

X\_train = dataset.iloc[:,:-1]

y\_train = dataset.iloc[:,-1:]

X\_test = dataset.iloc[:,1:]

print(X\_train.shape, y\_train.shape, X\_test.shape)

Subset data for modelling

52.from sklearn.linear\_model import LinearRegression

regression = LinearRegression()

regression.fit(X\_train, y\_train)

Linear regression code

53.Prediction

prediction = regression.predict(X\_test)

54.lasso regression

from sklearn import linear\_model

clf = linear\_model.Lasso(alpha=0.00001)

clf.fit(X\_train,y\_train)

predictions\_train = clf.predict(X\_train)

Predictions\_train

55.clf.score(X\_train, y\_train)

Return the coefficient of determination of the prediction.

Score data and calculate

56.predict lasso

predictions = clf.predict(X\_test)

Predictions

57.parse date

df1 = pd.read\_csv('sales\_train.csv',parse\_dates=[0])

from datetime import datetime

f1 = '%d.%m.%Y'

my\_parser = lambda date: pd.datetime.strptime(date, f1)

df1m = pd.read\_csv('sales\_train.csv', index\_col=[0], parse\_dates=[0], date\_parser=my\_parser)

df1m=df1m.reset\_index()

58.rename columns

DF4=DF2.rename(columns={"date":"ds","item\_cnt\_day":"y"})

DF4=DF2.rename(columns={"date":"ds","item\_cnt\_day":"y"}).groupby(["ds"]).sum()["y"].reset\_index()

59.plot data

plt.plot(DF5["ds"],DF5["y"])

60.FILTER by date time

mday=pd.to\_datetime("2015-08-01")

train\_index=DF5["ds"]<mday

test\_index=DF5["ds"]>=mday

x\_train=DF5[train\_index]

x\_test=DF5[test\_index]

dates\_test=DF5["ds"][test\_index]

61.conda install -c conda-forge fbprophet -y

pip install --upgrade plotly

Prophet algorithm install

62.plt.plot(DF5["ds"],DF5["y"])

63.from fbprophet import Prophet

m1=Prophet(yearly\_seasonality=True,weekly\_seasonality=True,

daily\_seasonality=False,

seasonality\_mode="multiplicative")

m1.fit(x\_train)

Prophet algorithm

64.future1=m1.make\_future\_dataframe(periods=92,freq="D")

display(future1.head())

display(future1.tail())

65.predict data using prophet

fcst1=m1.predict(future1)

fig=m1.plot\_components(fcst1)

plt.show()

66.rsquare values

ypred1=fcst1[-92:][["yhat"]].values

ytest1=x\_test["y"].values

from sklearn.metrics import r2\_score

score=r2\_score(ytest1,ypred1)

print(f'R2 score:{score:.4f}')

67.plot data timeseries

import matplotlib.dates as mdates

fig,ax=plt.subplots(figsize=(8,4))

ax.plot(dates\_test,ytest1,label="actual",c="k")

ax.plot(dates\_test,ypred1,label="predict",c="b")

weeks=mdates.WeekdayLocator(byweekday=mdates.TH)

ax.xaxis.set\_major\_locator(weeks)

ax.tick\_params(axis="x",rotation=90)

ax.grid()

ax.legend()

ax.set\_title("Result actual vs predict")

plt.show()

68.install libraries

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import mean\_squared\_error

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.linear\_model import Lasso

from sklearn.linear\_model import Lasso

from xgboost import XGBRegressor

69.number of unique values

item\_set["item\_name"].nunique()

70.information about dataset data type

train\_set.info()

71,stats about data

train\_set.describe()

72.missing values summation

train\_set.isnull().sum()

73.histogram of al train columns

train\_set.hist(figsize=(15,15), bins=6)

plt.show()

74.histogram of a particular column

train\_set['item\_cnt\_day'].hist(range=[-1, 10], facecolor='green', align='mid')

plt.show()

75.describe one column

train\_set['item\_cnt\_day'].describe()

76.correlation on train data

correlation\_matrix = train\_set.corr()

correlation\_matrix['item\_cnt\_day'].sort\_values(ascending = False)

correlation\_num = 6

correlation\_cols = correlation\_matrix.nlargest(correlation\_num,'item\_cnt\_day')['item\_cnt\_day'].index

correlation\_mat\_sales = np.corrcoef(train\_set[correlation\_cols].values.T)

sns.set(font\_scale=1.25)

f, ax = plt.subplots(figsize=(12, 9))

hm = sns.heatmap(correlation\_mat\_sales, cbar=True, annot=True, square=True, fmt='.2f', annot\_kws={'size': 7}, yticklabels=correlation\_cols.values, xticklabels=correlation\_cols.values)

plt.show()

77.Number of columns

print(len(x.columns))

78.train test split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x,y, test\_size = 0.3, random\_state = 60,shuffle=True)

print(len(X\_train))

print(len(X\_test))

79.predict and validation rmse mse ,

linear\_model = LinearRegression()

linear\_model.fit(X\_train, Y\_train)

#make predictions using linear regression

price\_predict =(linear\_model.predict(X\_test))

#Measure Performance

linear\_mse = mean\_squared\_error(Y\_test,price\_predict)

print("Mean Squared error: ",linear\_mse)

linear\_SqMse = np.sqrt(linear\_mse)

print("Root Mean Squared error: ", linear\_SqMse)

print("Explain variance score for Linear Regression =", round(sm.explained\_variance\_score(Y\_test,price\_predict),2))

print("R2 score for Linear Regression =", round(sm.r2\_score(Y\_test, price\_predict), 2))

80.lasso model

lasso\_model= Lasso(max\_iter=5000)

# Train the model with training data

lasso\_model.fit(X\_train,Y\_train)

#make predictions

price\_predict =(lasso\_model.predict(X\_test))

lasso\_mse = mean\_squared\_error(Y\_test,price\_predict)

print("Mean Squared Error : " , lasso\_mse)

lasso\_SqMse = np.sqrt(lasso\_mse)

print("Root Mean Squared error: ", lasso\_SqMse)

print("Explain variance score for Lasso Regression =", round(sm.explained\_variance\_score(Y\_test,price\_predict), 2))

print("R2 score for Lasso Regression =", round(sm.r2\_score(Y\_test, price\_predict), 2))

81.random forest regression

# Create a model with 50 decision trees

forest\_model= RandomForestRegressor(n\_estimators = 50, random\_state = 42)

# Train the model with training data

forest\_model.fit(X\_train,Y\_train)

#make predictions using random forest model

price\_predict =(forest\_model.predict(X\_test))

forest\_mse = mean\_squared\_error(Y\_test,price\_predict)

print("Mean Squared Error : " , forest\_mse)

forest\_SqMse = np.sqrt(forest\_mse)

print("Root Mean Squared error: ", forest\_SqMse)

print("Explain variance score for Random Forest Regression =", round(sm.explained\_variance\_score(Y\_test,price\_predict), 2))

print("R2 score for Random Forest Regression =", round(sm.r2\_score(Y\_test, price\_predict), 2))

82.Xgboost regression

# Create a model with 50 decision trees

xgbr= XGBRegressor()

# Train the model with training data

xgbr.fit(X\_train,Y\_train)

#make predictions using random forest model

price\_predict =(xgbr.predict(X\_test))

xgbr\_mse = mean\_squared\_error(Y\_test,price\_predict)

print("Mean Squared Error : " , xgbr\_mse)

xgbr\_SqMse = np.sqrt(xgbr\_mse)

print("Root Mean Squared error: ", xgbr\_SqMse)

print("Explain variance score for XGB Regression =", round(sm.explained\_variance\_score(Y\_test,price\_predict), 2))

print("R2 score for XGB Regression =", round(sm.r2\_score(Y\_test, price\_predict), 2))

83.print data with blanks

print(sales.head())

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

print(sales.info())

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

print(sales.describe())

84.filter data basis another dataset columns

sales = sales[sales['shop\_id'].isin(test['shop\_id'].unique())]

85.

Regression model building and output

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

#fitting data

regressor.fit(X\_train, Y\_train)

from sklearn.metrics import mean\_squared\_error

print('Train set mse:', mean\_squared\_error(Y\_train, regressor.predict(X\_train)))

print('Test set mse:', mean\_squared\_error(Y\_test, regressor.predict(X\_test)))

print('Test set score:', regressor.score(X\_train,Y\_train))

86.error for lstm

]

NotImplementedError: Cannot convert a symbolic Tensor (lstm\_1/strided\_slice:0) to a numpy array. This error may indicate that you're trying to pass a Tensor to a NumPy call, which is not supported

pip install -U numpy==1.18.5

And restart python

87.LSTM Model layers

import keras

from keras.models import Sequential

from keras.layers import LSTM,Dense,Dropout

from keras.models import load\_model, Model

# defining model

model = Sequential()

model.add(LSTM(units = 128,return\_sequences=True,input\_shape = (33,1)))

model.add(Dropout(0.5))

model.add(LSTM(units = 64,return\_sequences=False,activation='relu'))

# model.add(Dropout(0.5))

model.add(Dense(32,activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(16))

model.add(Dense(1))

# opt = keras.optimizers.Adam(learning\_rate=0.0001)

model.compile(loss = 'mse',optimizer = 'Nadam', metrics = ['mean\_squared\_error'])

model.summary()

#Splitting data for LSTM's

from sklearn.model\_selection import train\_test\_split

X\_train = np.expand\_dims(sales\_data.values[:,:-1],axis = 2)

y\_train = sales\_data.values[:,-1:]

model.fit(X\_train,y\_train, batch\_size = 1024,epochs = 10, validation\_split=0.1)

import matplotlib.pyplot as plt

plt.plot(model.history.history['loss'], label='Train loss')

plt.plot(model.history.history['val\_loss'], label='Validation loss')

plt.legend(loc='best')

# plt.title

plt.title('Regular LSTM')

plt.xlabel('Epochs')

plt.ylabel('MSE')

train\_pred = model.predict(X\_train)

87.Creating time series splits

tscv = TimeSeriesSplit(n\_splits=3, test\_size=2, gap=2)

88.neg\_mean\_squared\_error sis mean squared error

You are right, neg\_mean\_squared\_error is simple -1 \* mean\_squared\_error

89.In which environment is python executing

import sys

sys.executable

90.activate virtual env

.myenv\Scripts\activate.bat

91.remove duplicate data

df\_unique = df.drop\_duplicates(subset=['Pet', 'Color'])

92.df.replace(to\_replace ="Boston Celtics",

value ="Omega Warrior")

Replace data in columns

Recode data

Rename categories in a data

93.Histogram

import pandas as pd

my\_data = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 8, 6, 4, 2])

my\_data.hist()

94.Apologies for the confusion. In Windows, you can use the following command to activate a Conda environment:

```shell

conda activate <environment\_name>

```

If you encounter any issues with the `conda activate` command, you might need to ensure that Conda is properly installed and configured on your system. Additionally, make sure you are running the command in a terminal that has Conda initialized (e.g., the "Anaconda Prompt" or a terminal where you have executed `conda init`).

If you prefer an alternative to Conda, you can use `virtualenv` combined with `virtualenvwrapper-win` to create and manage virtual environments. Here's an example:

1. Install `virtualenv` and `virtualenvwrapper-win`:

```shell

pip install virtualenv virtualenvwrapper-win

```

2. Create a virtual environment:

```shell

mkvirtualenv myenv

```

3. Activate the virtual environment:

```shell

workon myenv

```

These commands allow you to create and activate virtual environments similar to Conda.

Powershell to move a item from one path for another

C:\Projects\Searchengine\imagetotext> Move-Item -Path C:\Users\user\Downloads\healthy-catfish-350100-53f09d76f28c.json -Destination C:\Projects\Searchengine\imagetotext\

Poweshell to convert 7z file and save in local

& "C:\Program Files\7-Zip\7z.exe" x -o"$destinationPath" "$sourceFile"

git clone <https://github.com/phil19851/Pythonprojects.git>

To clone a git

And go to to the git directory in local and then

Git pull

C:\Projects\Pythoncodes>git branch

\* master

C:\Projects\Pythoncodes>git branch Pythoncodes

C:\Projects\Pythoncodes>git checkout Pythoncodes

Switched to branch 'Pythoncodes'

C:\Projects\Pythoncodes>git push -u origin22 Pythoncodes

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 500.15 KiB | 9.81 MiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0