Final Project: Package Delivery Prediction Models

Ye Yuan (N18357924)

As the title shows, in this project I would like to:

1. Based on the given dataset, using two samples hypothesis test to show the differences of "On Time", "Missed" rates and "Shipment Days" between two countries: China and USA.
2. Using GridSearch method of Python to construct probability prediction models to estimate the 'On Time' probability after given the shipment time prediction and estimate the 'Missed' probability, helping customers to buy proper insurance.
3. Construct neural network to predict the numerical variable "Shipment Days".

The dataset link is: https://data.world/usaid/supply-chain-shipment-pricing.
The overall project coding procedures are:

1. Data Preparation (Fill Missing Values, One-hot Encoding, Feature Selections)
2. General Data Summary
3. Two Sample Hypothesis Test
4. 'On Time' Probability Prediction Model (Logistics Regression, Classification Tree)
5. 'Missed' Probability Prediction Model (Logistics Regression, Classification Tree)
6. 'Shipment Days' Prediction Model (Neural Network)
7. Final Model Evaluation

```
In [1]: import pandas as pd
         import numpy as np
         import random

In [2]: dataset = pd.DataFrame(pd.read_csv('./SCMS_Delivery_History_Dataset.csv'))

In [3]: dataset.shape

Out[3]: (10324, 33)

In [4]: dataset_ori['On Time'] = dataset['Delivered to Client Date']
                ==dataset['Scheduled Delivery Date']

In [5]: dataset_china = dataset_ori[dataset_ori['Country']=='China']
         dataset_US = dataset_ori[dataset_ori['Country']=='US']
```

In [6]: dataset_method.mean()

Out[6]:

Shipment Mode	ID	Unit of Measure (Per Pack)	Line Item Quantity
Air	44947.163586	82.342549	9143.640275
Air Charter	75009.315385	55.415385	45352.953846
Ocean	47649.401617	71.832884	65314.692722
Truck	64488.961837	72.593640	26803.161484

Shipment Mode	Line Item Value	Pack Price	Unit Price
Air	102615.072379	27.984085	0.816660
Air Charter	379034.201769	8.638462	0.187400
Ocean	340102.274367	6.568949	0.152480
Truck	207555.607456	13.230212	0.251201

Shipment Mode	Line Item Insurance (USD)	On Time
Air	162.401546	0.710453
Air Charter	543.556585	0.189231
Ocean	567.808518	0.784367
Truck	294.259442	0.437456

In [7]: dataset_China.shape

Out[7]: (4722, 36)

In [8]: dataset_China.head()

Out[8]:

	ID	Project Code	PQ #	PO / SO #	ASN/DN #	Country	Managed By
1	3	108-VN-T01	Pre-PQ Process	SCMS-13	ASN-85	China	PMO - US
3	15	108-VN-T01	Pre-PQ Process	SCMS-78	ASN-50	China	PMO - US
4	16	108-VN-T01	Pre-PQ Process	SCMS-81	ASN-55	China	PMO - US
6	44	110-ZM-T01	Pre-PQ Process	SCMS-139	ASN-130	China	PMO - US
12	62	102-NG-T01	Pre-PQ Process	SCMS-230	ASN-144	China	PMO - US

	Fulfill Via	Vendor INCO Term	Shipment Mode	...	Pack Price
1	Direct Drop	EXW	Air	...	6.20
3	Direct Drop	EXW	Air	...	3.99
4	Direct Drop	EXW	Air	...	3.20
6	Direct Drop	DDU	Air	...	32.40
12	Direct Drop	EXW	Air	...	85.00

	Unit Price	Manufacturing Site	First Line Designation
1	0.03	Aurobindo Unit III, India	Yes
3	0.07	Ranbaxy, Paonta Shahib, India	Yes
4	0.05	Aurobindo Unit III, India	Yes
6	0.36	MSD South Granville Australia	Yes
12	0.85	EY Laboratories, USA	Yes

```
                Weight (Kilograms)                  Freight Cost (USD)  \
1                              358                             4521.5
3                             1855                            16007.06
4                             7590                            45450.08
6                              328  Freight Included in Commodity Cost
12  Weight Captured Separately                  Invoiced Separately

    Line Item Insurance (USD) On Time Missed Shipment Days
1                         NaN       1      0             3
3                         NaN       1      0             5
4                         NaN       1      0             3
6                         NaN       1      0             3
12                        NaN       1      0             2

[5 rows x 36 columns]
```

```
In [9]: dataset_US_.shape
```

```
Out[9]: (5242, 36)
```

```
In [10]: dataset_China.shape
```

```
Out[10]: (4722, 36)
```

```
In [11]: dataset_China.columns
```

```
Out[11]: Index(['ID', 'Project Code', 'PQ #', 'PO / SO #', 'ASN/DN #', 'Country',
                'Managed By', 'Fulfill Via', 'Vendor INCO Term', 'Shipment Mode',
                'PQ First Sent to Client Date', 'PO Sent to Vendor Date',
                'Scheduled Delivery Date', 'Delivered to Client Date',
                'Delivery Recorded Date', 'Product Group', 'Sub Classification',
                'Vendor', 'Item Description', 'Molecule/Test Type', 'Brand', 'Dosage',
                'Dosage Form', 'Unit of Measure (Per Pack)', 'Line Item Quantity',
                'Line Item Value', 'Pack Price', 'Unit Price', 'Manufacturing Site',
                'First Line Designation', 'Weight (Kilograms)', 'Freight Cost (USD)',
                'Line Item Insurance (USD)', 'On Time', 'Missed', 'Shipment Days'],
               dtype='object')
```

```
In [12]: dataset_US.columns
```

```
Out[12]: Index(['ID', 'Project Code', 'PQ #', 'PO / SO #', 'ASN/DN #', 'Country',
                'Managed By', 'Fulfill Via', 'Vendor INCO Term', 'Shipment Mode',
                'PQ First Sent to Client Date', 'PO Sent to Vendor Date',
                'Scheduled Delivery Date', 'Delivered to Client Date',
                'Delivery Recorded Date', 'Product Group', 'Sub Classification',
                'Vendor', 'Item Description', 'Molecule/Test Type', 'Brand', 'Dosage',
                'Dosage Form', 'Unit of Measure (Per Pack)', 'Line Item Quantity',
                'Line Item Value', 'Pack Price', 'Unit Price', 'Manufacturing Site',
```

```
       'First Line Designation', 'Weight (Kilograms)', 'Freight Cost (USD)',
       'Line Item Insurance (USD)', 'On Time', 'Missed', 'Shipment Days'],
      dtype='object')
```

In [13]:
```
dataset_China_sim = dataset_China
       [['Country','Shipment Mode','Unit of Measure (Per Pack)','Line Item Quantity',
       'Pack Price','Weight (Kilograms)','Freight Cost (USD)','On Time', 'Missed',
       'Shipment Days']]
dataset_US_sim = dataset_US
       [['Country','Shipment Mode','Unit of Measure (Per Pack)','Line Item Quantity',
        'Pack Price','Weight (Kilograms)','Freight Cost (USD)','On Time', 'Missed',
        'Shipment Days']]
dataset_China_sim.head()
```

Out[13]:

	Country	Shipment Mode	Unit of Measure (Per Pack)	Line Item Quantity	\
1	China	Air	240	1000	
3	China	Air	60	31920	
4	China	Air	60	38000	
6	China	Air	90	135	
12	China	Air	100	10	

	Pack Price	Weight (Kilograms)	\
1	6.20	358	
3	3.99	1855	
4	3.20	7590	
6	32.40	328	
12	85.00	Weight Captured Separately	

	Freight Cost (USD)	On Time	Missed	Shipment Days
1	4521.5	1	0	3
3	16007.06	1	0	5
4	45450.08	1	0	3
6	Freight Included in Commodity Cost	1	0	3
12	Invoiced Separately	1	0	2

In [14]:
```
value1 = np.mean(dataset_China_sim['Freight Cost (USD)'].
       convert_objects(convert_numeric=True)/dataset_China_sim['Line Item Quantity'])
transit = dataset_China_sim['Freight Cost (USD)'].
       convert_objects(convert_numeric=True)/dataset_China_sim['Line Item Quantity']
transit1 = transit.fillna(value1)

value2 = np.mean(dataset_China_sim['Weight (Kilograms)'].
        convert_objects(convert_numeric=True)/dataset_China_sim['Line Item Quantity'])
transit = dataset_China_sim['Weight (Kilograms)'].
        convert_objects(convert_numeric=True)/dataset_China_sim['Line Item Quantity']
transit2 = transit.fillna(value2)
```

In [15]:
```
data_China = dataset_China_sim.drop(['Country'],axis=1)
data_China['Freight Cost (USD)'] = transit1
data_China['Weight (Kilograms)'] = transit2
```

```
In [16]: data_China['Encode1'] = np.zeros(data_China.shape[0])
         data_China['Encode2'] = np.zeros(data_China.shape[0])

In [17]: data_China1 = data_China[data_China['Shipment Mode']=='Air']
         data_China1['Encode1'] = np.ones(data_China1.shape[0])

In [18]: data_China2 = data_China[data_China['Shipment Mode']=='Air Charter']
         data_China2['Encode1'] = np.ones(data_China2.shape[0])
         data_China2['Encode2'] = np.ones(data_China2.shape[0])

In [19]: data_China4 = data_China[data_China['Shipment Mode']=='Truck']
         data_China4['Encode2'] = np.ones(data_China4.shape[0])

In [20]: data_China = data_China1.append(data_China2)
         data_China = data_China.append(data_China3)
         data_China = data_China.append(data_China4)

In [21]: data_China.head()
```

Out[21]:

	Shipment Mode	Unit of Measure (Per Pack)	Line Item Quantity	Pack Price	\
1	Air	240	1000	6.20	
3	Air	60	31920	3.99	
4	Air	60	38000	3.20	
6	Air	90	135	32.40	
12	Air	100	10	85.00	

	Weight (Kilograms)	Freight Cost (USD)	On Time	Missed	Shipment Days	\
1	0.358000	4.521500	1	0	3	
3	0.058114	0.501474	1	0	5	
4	0.199737	1.196055	1	0	3	
6	2.429630	35.186846	1	0	3	
12	6.200352	35.186846	1	0	2	

	Encode1	Encode2
1	1.0	0.0
3	1.0	0.0
4	1.0	0.0
6	1.0	0.0
12	1.0	0.0

```
In [22]: dataset_US_sim['On Time'] = dataset_US_sim['On Time'].astype('int')
         dataset_US_sim.head()
```

Out[22]:

	Country	Shipment Mode	Unit of Measure (Per Pack)	Line Item Quantity	\
0	US	Air	30	19	
2	US	Air	100	500	
5	US	Air	240	416	
7	US	Air	60	16667	
8	US	Air	60	273	

```
   Pack Price     Weight (Kilograms)     Freight Cost (USD)  On Time  Missed  \
0       29.00                     13                 780.34        1       0
2       80.00                    171                1653.78        1       0
5        5.35                    504                5920.42        1       0
7        3.65                   1478                6212.41        1       0
8        1.95  See ASN-93 (ID#:1281)  See ASN-93 (ID#:1281)        1       0

   Shipment Days
0              4
2              3
5              3
7              2
8              5
```

In [23]:
```
value1 = np.mean(dataset_US_sim['Freight Cost (USD)'].
       convert_objects(convert_numeric=True)/dataset_US_sim['Line Item Quantity'])
transit = dataset_US_sim['Freight Cost (USD)'].
        convert_objects(convert_numeric=True)/dataset_US_sim['Line Item Quantity']
transit1 = transit.fillna(value1)

value2 = np.mean(dataset_US_sim['Weight (Kilograms)'].
        convert_objects(convert_numeric=True)/dataset_US_sim['Line Item Quantity'])
transit = dataset_US_sim['Weight (Kilograms)'].
        convert_objects(convert_numeric=True)/dataset_US_sim['Line Item Quantity']
transit2 = transit.fillna(value2)
```

In [24]:
```
data_US = dataset_US_sim.drop(['Country'],axis=1)
data_US['Freight Cost (USD)'] = transit1
data_US['Weight (Kilograms)'] = transit2
```

In [25]:
```
data_US['Encode1'] = np.zeros(data_US.shape[0])
data_US['Encode2'] = np.zeros(data_US.shape[0])
```

In [26]:
```
data_US1 = data_US[data_US['Shipment Mode']=='Air']
data_US1['Encode1'] = np.ones(data_US1.shape[0])

data_US2 = data_US[data_US['Shipment Mode']=='Air Charter']
data_US2['Encode1'] = np.ones(data_US2.shape[0])
data_US2['Encode2'] = np.ones(data_US2.shape[0])

data_US3 = data_US[data_US['Shipment Mode']=='Ocean']

data_US4 = data_US[data_US['Shipment Mode']=='Truck']
data_US4['Encode2'] = np.ones(data_US4.shape[0])
```

In [27]:
```
data_US = data_US1.append(data_US2)
data_US = data_US.append(data_US3)
data_US = data_US.append(data_US4)
```

In [28]:
```
data_US.head()
```

Out[28]:

	Shipment Mode	Unit of Measure (Per Pack)	Line Item Quantity	Pack Price	\
0	Air	30	19	29.00	
2	Air	100	500	80.00	
5	Air	240	416	5.35	
7	Air	60	16667	3.65	
8	Air	60	273	1.95	

	Weight (Kilograms)	Freight Cost (USD)	On Time	Missed	Shipment Days	\
0	0.684211	41.070526	1	0	4	
2	0.342000	3.307560	1	0	3	
5	1.211538	14.231779	1	0	3	
7	0.088678	0.372737	1	0	2	
8	3.572041	22.507133	1	0	5	

	Encode1	Encode2
0	1.0	0.0
2	1.0	0.0
5	1.0	0.0
7	1.0	0.0
8	1.0	0.0

In [29]:
```
data_China_group = data_China.drop(['Unit of Measure (Per Pack)',
      'Line Item Quantity'],axis=1)
data_China_group = data_China_group.groupby(['Shipment Mode'])
data_China_group.mean()
```

Out[29]:

Shipment Mode	Pack Price	Weight (Kilograms)	Freight Cost (USD)	On Time	\
Air	28.427124	4.174945	37.736487	0.800910	
Air Charter	8.335083	38.698004	89.592059	0.316832	
Ocean	7.120561	1.631347	9.254075	1.000000	
Truck	12.977355	3.883393	21.495396	0.494505	

Shipment Mode	Missed	Shipment Days	Encode1	Encode2
Air	0.001400	3.098670	1.0	0.0
Air Charter	0.003300	2.122112	1.0	1.0
Ocean	0.010204	5.280612	0.0	0.0
Truck	0.000000	3.301099	0.0	1.0

In [30]:
```
from scipy import stats

t_test,p_value_t = stats.ttest_ind(data_China['Shipment Days'],
        data_US['Shipment Days'],equal_var=False)
print(p_value_t)
```

```
2.74880945502727e-37
```

```
In [31]: def ztest_proportion_two_samples(x1, n1, x2, n2, one_sided=False):
             p1 = x1/n1
             p2 = x2/n2

             p = (x1+x2)/(n1+n2)
             se = p*(1-p)*(1/n1+1/n2)
             se = np.sqrt(se)

             z = (p1-p2)/se
             p = 1-stats.norm.cdf(abs(z))
             p *= 2-one_sided # if not one_sided: p *= 2
             return z, p

In [32]: n1_ontime = data_China.shape[0]
         n2_ontime = np.sum(data_China['On Time'])
         n3_ontime = data_US.shape[0]
         n4_ontime = np.sum(data_US['On Time'])

         z_test_ontime,p_value_ontime = ztest_proportion_two_samples(n1_ontime,
                 n2_ontime,n3_ontime,n4_ontime,one_sided=True)
         print(p_value_ontime)
```

```
1.45726499153682e-05
```

```
In [33]: n1_missed = data_China.shape[0]
         n2_missed = np.sum(data_China['Missed'])
         n3_missed = data_US.shape[0]
         n4_missed = np.sum(data_US['Missed'])

         z_test_missed,p_value_missed = ztest_proportion_two_samples(n1_missed,
                 n2_missed,n3_missed,n4_missed,one_sided=True)
         print(p_value_missed)
```

```
0.01674819116453
```

```
In [34]: data_US_group = data_US.drop(['Unit of Measure (Per Pack)','Line Item Quantity'],
                axis=1)
         data_US_group = data_US_group.groupby(['Shipment Mode'])
         data_US_group.mean()
```

Out[34]:

Shipment Mode	Pack Price	Weight (Kilograms)	Freight Cost (USD)	On Time
Air	27.595081	3.577714	25.896660	0.709370
Air Charter	8.903372	8.939229	30.276819	0.178674
Ocean	5.951143	1.042227	7.789781	0.794286
Truck	13.465809	2.590360	14.893855	0.446416

```
                    Missed  Shipment Days  Encode1  Encode2
Shipment Mode
Air               0.002765       3.349002      1.0      0.0
Air Charter       0.000000       2.322767      1.0      1.0
Ocean             0.000000       5.371429      0.0      0.0
Truck             0.000000       3.800683      0.0      1.0
```

```
In [35]: China_features = data_China.drop(['Shipment Mode','On Time',
               'Missed','Shipment Days'],axis=1)
         China_ontime = data_China['On Time']
         China_missed = data_China['Missed']
         China_days = data_China['Shipment Days']
```

```
In [36]: US_features = data_US.drop(['Shipment Mode','On Time','Missed','Shipment Days'],
               axis=1)
         US_ontime = data_US['On Time']
         US_missed = data_US['Missed']
         US_days = data_US['Shipment Days']
```

```
In [37]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import roc_auc_score, confusion_matrix
```

```
In [38]: CXtr, CXts, Cytr, Cyts = train_test_split(China_features, China_ontime,
               test_size=0.33, random_state=0)
```

```
In [39]: Cmodel_ontime_log = LogisticRegression(random_state=0, C=10E6)
         Cmodel_ontime_log.fit(CXtr, Cytr)

         Cyhat = Cmodel_ontime_log.predict(CXts)
         con = confusion_matrix(Cyts, Cyhat, labels=(0,1))
         acc = (con[0,0]+con[1,1])/np.sum(con)
         sen = con[1,1]/(con[1,0]+con[1,1])
         spe = con[0,0]/(con[0,0]+con[0,1])
         pre = con[1,1]/(con[1,1]+con[0,1])
         f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
         print("The accuracy measurements of the classification tree are: ")
         print("Accuracy = %.4f" %acc)
         print("Sensitivity = %.4f" %sen)
         print("Specificity = %.4f" %spe)
         print("Precision = %.4f" %pre)
         print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.7691
Sensitivity = 0.8831
Specificity = 0.5135
Precision = 0.8027
F1 Score = 0.8410
```

```
In [40]: from sklearn import tree
         from sklearn.model_selection import GridSearchCV

In [41]: from sklearn import tree
         from sklearn.model_selection import GridSearchCV

         parameters = {'criterion':('gini', 'entropy'),
                       'min_samples_split':[2,3,4,5],
                       'max_depth':[9,10,11,12],
                       'class_weight':('balanced', None),
                       'presort':(False,True),
                      }

         #Model Construction
         tr = tree.DecisionTreeClassifier()
         gsearch = GridSearchCV(tr, parameters)
         gsearch.fit(CXtr, Cytr)
         Cmodel_ontime_tree = gsearch.best_estimator_

         score = Cmodel_ontime_tree.score(CXts, Cyts)
         print("The accuracy is: %.4f" %score)
```

```
The accuracy is: 0.8005
```

```
In [42]: Cyhat = Cmodel_ontime_tree.predict(CXts)
         con = confusion_matrix(Cyts, Cyhat, labels=(0,1))
         acc = (con[0,0]+con[1,1])/np.sum(con)
         sen = con[1,1]/(con[1,0]+con[1,1])
         spe = con[0,0]/(con[0,0]+con[0,1])
         pre = con[1,1]/(con[1,1]+con[0,1])
         f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
         print("The accuracy measurements of the classification tree are: ")
         print("Accuracy = %.4f" %acc)
         print("Sensitivity = %.4f" %sen)
         print("Specificity = %.4f" %spe)
         print("Precision = %.4f" %pre)
         print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.8005
Sensitivity = 0.8924
Specificity = 0.5946
Precision = 0.8315
F1 Score = 0.8609
```

```
In [43]: UXtr, UXts, Uytr, Uyts = train_test_split(US_features, US_ontime,
                test_size=0.33, random_state=0)
```

```
In [44]: Umodel_ontime_log = LogisticRegression(random_state=0, C=10E6)
         Umodel_ontime_log.fit(UXtr, Uytr)

         Uyhat = Umodel_ontime_log.predict(UXts)
         con = confusion_matrix(Uyts, Uyhat, labels=(0,1))
         acc = (con[0,0]+con[1,1])/np.sum(con)
         sen = con[1,1]/(con[1,0]+con[1,1])
         spe = con[0,0]/(con[0,0]+con[0,1])
         pre = con[1,1]/(con[1,1]+con[0,1])
         f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
         print("The accuracy measurements of the classification tree are: ")
         print("Accuracy = %.4f" %acc)
         print("Sensitivity = %.4f" %sen)
         print("Specificity = %.4f" %spe)
         print("Precision = %.4f" %pre)
         print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.7150
Sensitivity = 0.8680
Specificity = 0.4979
Precision = 0.7105
F1 Score = 0.7814
```

```
In [45]: parameters = {'criterion':('gini', 'entropy'),
                        'min_samples_split':[2,3,4,5],
                        'max_depth':[9,10,11,12],
                        'class_weight':('balanced', None),
                        'presort':(False,True),
                       }

         #Model Construction
         tr = tree.DecisionTreeClassifier()
         gsearch = GridSearchCV(tr, parameters)
         gsearch.fit(UXtr, Uytr)
         Umodel_ontime_tree = gsearch.best_estimator_

         score = Umodel_ontime_tree.score(UXts, Uyts)
         print("The accuracy is: %.4f" %score)
```

```
The accuracy is: 0.7387
```

```
In [46]: Uyhat = Umodel_ontime_tree.predict(UXts)
         con = confusion_matrix(Uyts, Uyhat, labels=(0,1))
         acc = (con[0,0]+con[1,1])/np.sum(con)
         sen = con[1,1]/(con[1,0]+con[1,1])
         spe = con[0,0]/(con[0,0]+con[0,1])
```

```
pre = con[1,1]/(con[1,1]+con[0,1])
f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
print("The accuracy measurements of the classification tree are: ")
print("Accuracy = %.4f" %acc)
print("Sensitivity = %.4f" %sen)
print("Specificity = %.4f" %spe)
print("Precision = %.4f" %pre)
print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.7387
Sensitivity = 0.7813
Specificity = 0.6783
Precision = 0.7752
F1 Score = 0.7782
```

```
In [47]: CXtr_m, CXts_m, Cytr_m, Cyts_m = train_test_split(China_features,
                China_missed, test_size=0.33, random_state=0)
```

```
In [48]: Cmodel_missed_log = LogisticRegression(random_state=0, C=10E6)
         Cmodel_missed_log.fit(CXtr_m, Cytr_m)

         Cyhat_m = Cmodel_missed_log.predict(CXts_m)
         con = confusion_matrix(Cyts_m, Cyhat_m, labels=(0,1))
         acc = (con[0,0]+con[1,1])/np.sum(con)
         sen = con[1,1]/(con[1,0]+con[1,1])
         spe = con[0,0]/(con[0,0]+con[0,1])
         pre = con[1,1]/(con[1,1]+con[0,1])
         f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
         print("The accuracy measurements of the classification tree are: ")
         print("Accuracy = %.4f" %acc)
         print("Sensitivity = %.4f" %sen)
         print("Specificity = %.4f" %spe)
         print("Precision = %.4f" %pre)
         print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.9981
Sensitivity = 0.0000
Specificity = 0.9994
Precision = 0.0000
F1 Score = 0.0000
```

```
In [49]: parameters = {'criterion':('gini', 'entropy'),
                        'min_samples_split':[2,3,4,5],
                        'max_depth':[9,10,11,12],
                        'class_weight':('balanced', None),
```

```
                        'presort':(False,True),
                       }

            #Model Construction
            tr = tree.DecisionTreeClassifier()
            gsearch = GridSearchCV(tr, parameters)
            gsearch.fit(CXtr_m, Cytr_m)
            Cmodel_missed_tree = gsearch.best_estimator_

            score = Cmodel_missed_tree.score(CXts_m, Cyts_m)
            print("The accuracy is: %.4f" %score)
```

```
The accuracy is: 0.9955
```

```
In [50]: Cyhat_m = Cmodel_missed_tree.predict(CXts_m)
           con = confusion_matrix(Cyts_m, Cyhat_m, labels=(0,1))
           acc = (con[0,0]+con[1,1])/np.sum(con)
           sen = con[1,1]/(con[1,0]+con[1,1])
           spe = con[0,0]/(con[0,0]+con[0,1])
           pre = con[1,1]/(con[1,1]+con[0,1])
           f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
           print("The accuracy measurements of the classification tree are: ")
           print("Accuracy = %.4f" %acc)
           print("Sensitivity = %.4f" %sen)
           print("Specificity = %.4f" %spe)
           print("Precision = %.4f" %pre)
           print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.9955
Sensitivity = 0.0000
Specificity = 0.9968
Precision = 0.0000
F1 Score = 0.0000
```

```
In [51]: UXtr_m, UXts_m, Uytr_m, Uyts_m = train_test_split(US_features, US_missed,
                 test_size=0.33, random_state=0)
```

```
In [52]: Umodel_missed_log = LogisticRegression(random_state=0, C=10E6)
           Umodel_missed_log.fit(UXtr_m, Uytr_m)

           Uyhat_m = Umodel_missed_log.predict(UXts_m)
           con = confusion_matrix(Uyts_m, Uyhat_m, labels=(0,1))
           acc = (con[0,0]+con[1,1])/np.sum(con)
           sen = con[1,1]/(con[1,0]+con[1,1])
           spe = con[0,0]/(con[0,0]+con[0,1])
           pre = con[1,1]/(con[1,1]+con[0,1])
```

```
            f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
            print("The accuracy measurements of the classification tree are: ")
            print("Accuracy = %.4f" %acc)
            print("Sensitivity = %.4f" %sen)
            print("Specificity = %.4f" %spe)
            print("Precision = %.4f" %pre)
            print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.9994
Sensitivity = 0.0000
Specificity = 1.0000
Precision = 0.0000
F1 Score = 0.0000
```

```
In [53]: parameters = {'criterion':('gini', 'entropy'),
                          'min_samples_split':[2,3,4,5],
                          'max_depth':[9,10,11,12],
                          'class_weight':('balanced', None),
                          'presort':(False,True),
                         }

            #Model Construction
            tr = tree.DecisionTreeClassifier()
            gsearch = GridSearchCV(tr, parameters)
            gsearch.fit(UXtr_m, Uytr_m)
            Umodel_missed_tree = gsearch.best_estimator_

            score = Umodel_missed_tree.score(UXts_m, Uyts_m)
            print("The accuracy is: %.4f" %score)
```

```
The accuracy is: 0.9954
```

```
In [54]: Uyhat_m = Umodel_missed_tree.predict(UXts_m)
            con = confusion_matrix(Uyts_m, Uyhat_m, labels=(0,1))
            acc = (con[0,0]+con[1,1])/np.sum(con)
            sen = con[1,1]/(con[1,0]+con[1,1])
            spe = con[0,0]/(con[0,0]+con[0,1])
            pre = con[1,1]/(con[1,1]+con[0,1])
            f1 = 2*con[1,1]/(2*con[1,1]+con[1,0]+con[0,1])
            print("The accuracy measurements of the classification tree are: ")
            print("Accuracy = %.4f" %acc)
            print("Sensitivity = %.4f" %sen)
            print("Specificity = %.4f" %spe)
            print("Precision = %.4f" %pre)
            print("F1 Score = %.4f" %f1)
```

```
The accuracy measurements of the classification tree are:
Accuracy = 0.9954
Sensitivity = 0.0000
Specificity = 0.9960
Precision = 0.0000
F1 Score = 0.0000
```

```
In [55]: from tensorflow.keras.models import Model, Sequential
         from tensorflow.keras.layers import Dense, Activation
         import tensorflow.keras.backend as K
         from tensorflow.keras import optimizers
```

```
In [56]: CXtr_d, CXts_d, Cytr_d, Cyts_d = train_test_split(China_features,
               China_days, test_size=0.33, random_state=0)
```

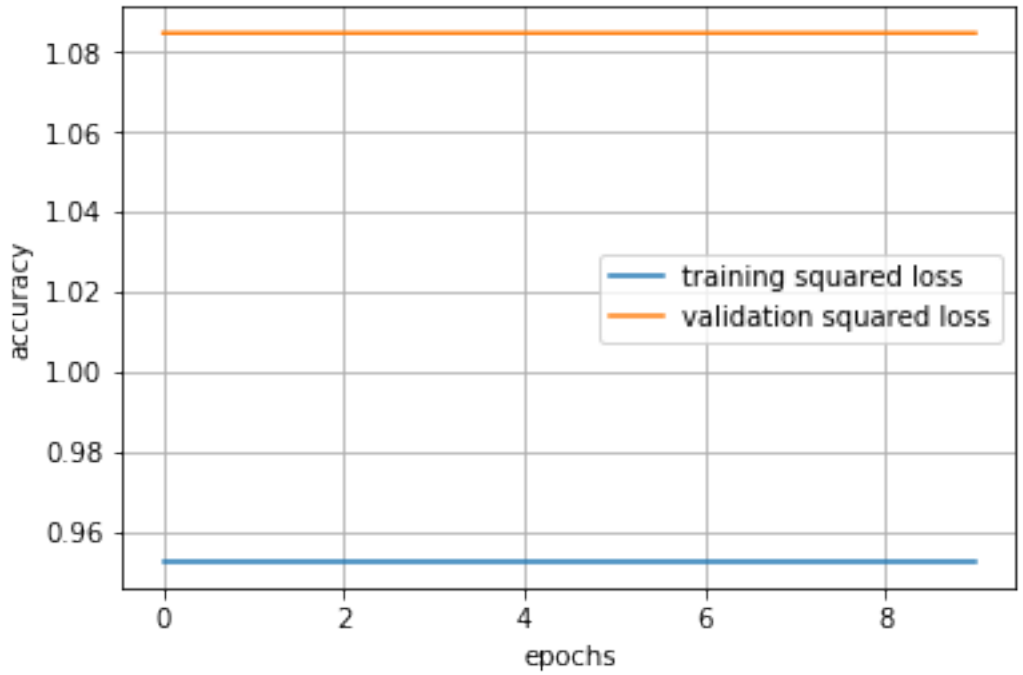
```
In [57]: from sklearn.preprocessing import scale

         CXtr_d_scaled = scale(CXtr_d, axis=0, with_mean=True, with_std=True, copy=True)
         CXts_d_scaled = scale(CXts_d, axis=0, with_mean=True, with_std=True, copy=True)
```

```
In [58]: K.clear_session()
         nin = CXtr_d.shape[1]
         nh = 256
         nout = 1
         Cmodel_neural = Sequential()
         Cmodel_neural.add(Dense(units=nh, input_shape=(nin,), activation='linear',
                 name='hidden'))
         Cmodel_neural.add(Dense(units=nout, activation='softmax', name='output'))

         Cmodel_neural.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden (Dense)               (None, 256)               2048
_________________________________________________________________
output (Dense)               (None, 1)                 257
=================================================================
Total params: 2,305
Trainable params: 2,305
Non-trainable params: 0
_________________________________________________________________
```

```
In [59]: opt = optimizers.Adam(lr=0.001)
         Cmodel_neural.compile(optimizer=opt,loss='mean_squared_error')

         hist = Cmodel_neural.fit(CXtr_d_scaled, Cytr_d, epochs=10, batch_size=100,
                 validation_data=(CXts_d_scaled,Cyts_d), verbose=0)
```

```
In [60]: hist = Cmodel_neural.fit(CXtr_d_scaled, Cytr_d, epochs=10, batch_size=100,
                validation_data=(CXts_d_scaled,Cyts_d), verbose=0)

In [61]: import matplotlib.pyplot as plt

         tr_loss = hist.history['loss']
         val_loss = hist.history['val_loss']
         plt.plot(tr_loss)
         plt.plot(val_loss)
         plt.grid()
         plt.xlabel('epochs')
         plt.ylabel('accuracy')
         plt.legend(['training squared loss', 'validation squared loss'])

Out[61]: <matplotlib.legend.Legend at 0x1a33fe8e80>
```

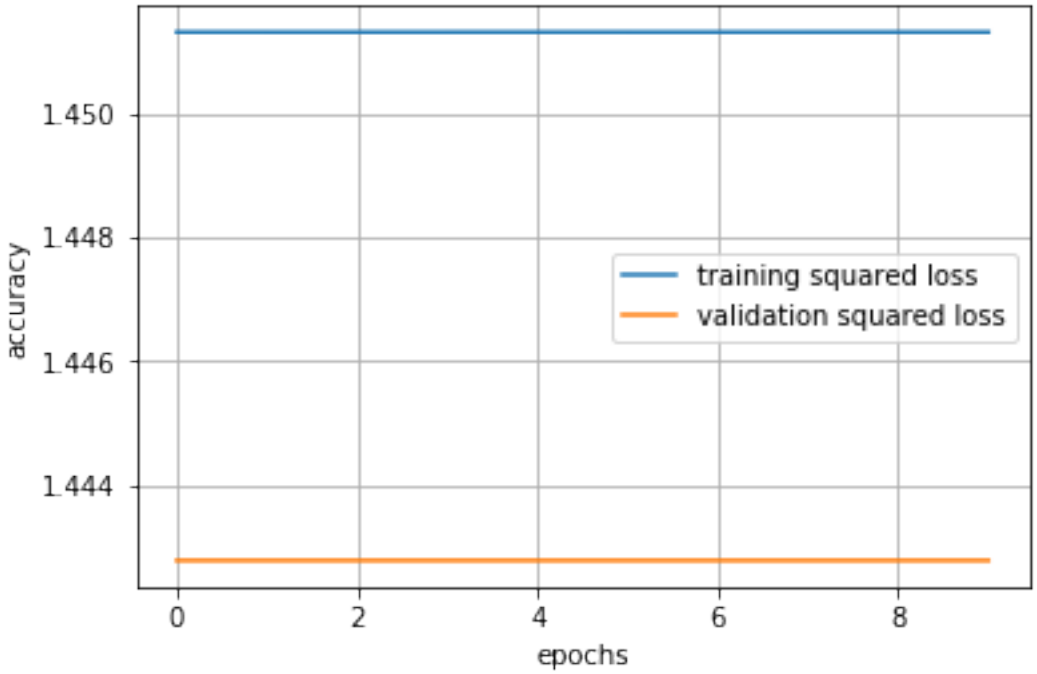



```
In [62]: UXtr_d, UXts_d, Uytr_d, Uyts_d = train_test_split(US_features,
                US_days, test_size=0.33, random_state=0)

In [63]: UXtr_d_scaled = scale(UXtr_d, axis=0, with_mean=True, with_std=True, copy=True)
         UXts_d_scaled = scale(UXts_d, axis=0, with_mean=True, with_std=True, copy=True)

In [64]: K.clear_session()

         nin = UXtr_d.shape[1]
```

```
nh = 256
nout = 1
Umodel_neural = Sequential()
Umodel_neural.add(Dense(units=nh, input_shape=(nin,), activation='linear',
        name='hidden'))
Umodel_neural.add(Dense(units=nout, activation='softmax', name='output'))

Umodel_neural.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden (Dense)               (None, 256)               2048
_________________________________________________________________
output (Dense)               (None, 1)                 257
=================================================================
Total params: 2,305
Trainable params: 2,305
Non-trainable params: 0
_________________________________________________________________
```

```
In [65]: opt = optimizers.Adam(lr=0.001)
         Umodel_neural.compile(optimizer=opt,loss='mean_squared_error')
         hist = Umodel_neural.fit(UXtr_d_scaled, Uytr_d, epochs=10, batch_size=100,
                 validation_data=(UXts_d_scaled,Uyts_d), verbose=0)

In [66]: tr_loss = hist.history['loss']
         val_loss = hist.history['val_loss']
         plt.plot(tr_loss)
         plt.plot(val_loss)
         plt.grid()
         plt.xlabel('epochs')
         plt.ylabel('accuracy')
         plt.legend(['training squared loss', 'validation squared loss'])

Out[66]: <matplotlib.legend.Legend at 0x1a34505fd0>
```




```
In [67]: test_China = np.array([[82.860742, 9490.364241, 28.427124,4.174945,37.736487,1.0,0.0],
                                [53.366337,47681.125413,8.335083,38.698004,89.592059,1.0,1.0],
                                [70.510204,58347.750000,7.120561,1.631347,9.254075,0.0,0.0],
                                [72.953846,26734.565568,12.977355,3.883393,21.495396,0.0,1.0]])
         test_US = np.array([[81.887558,8839.204916,27.595081,3.577714,25.896660,1.1,0.0],
                             [57.204611,43319.997118,8.903372,8.939229,30.276819,1.0,1.0],
                              [73.314286,73117.668571,5.951143,1.042227,7.789781,0.0,0.0],
                              [72.258020,26867.075085,13.465809,2.590360,14.893855,0.0,1.0]])

In [68]: China_label_log = Cmodel_ontime_log.predict(test_China)
         China_label_tree = Cmodel_ontime_tree.predict(test_China)
         China_probability = Cmodel_ontime_log.predict_proba(test_China)[:,1]
         print(China_label_log)
         print(China_label_tree)
         print(China_probability)

[1 1 1 1]
[1 1 1 1]
[0.93275542 0.94997342 0.87270544 0.5417225 ]


In [69]: China_label_log2 = Cmodel_missed_log.predict(test_China)
         China_label_tree2 = Cmodel_missed_tree.predict(test_China)
         China_probability2 = Cmodel_missed_log.predict_proba(test_China)[:,1]
         print(China_label_log2)
```

```
          print(China_label_tree2)
          print(China_probability2)
```

```
[0 0 0 0]
[0 0 0 0]
[6.90635094e-04 2.72430494e-07 1.01042995e-02 1.06783839e-03]
```

```
In [70]: US_label_log = Umodel_ontime_log.predict(test_US)
          US_label_tree = Umodel_ontime_tree.predict(test_US)
          US_probability = Umodel_ontime_log.predict_proba(test_US)[:,1]
          print(US_label_log)
          print(US_label_tree)
          print(US_probability)
```

```
[1 1 1 0]
[1 1 1 0]
[0.83427353 0.89669405 0.79995337 0.4596364 ]
```

```
In [71]: US_label_log2 = Umodel_missed_log.predict(test_US)
          US_label_tree2 = Umodel_missed_tree.predict(test_US)
          US_probability2 = Umodel_missed_log.predict_proba(test_US)[:,1]
          print(US_label_log2)
          print(US_label_tree2)
          print(US_probability2)
```

```
[0 0 0 0]
[0 0 0 0]
[1.64041385e-03 5.18040856e-05 9.34052110e-04 4.00104823e-04]
```

```
In [72]: test_China_scaled = scale(test_China, axis=0, with_mean=True, with_std=True, copy=True)
          test_US_scaled = scale(test_US, axis=0, with_mean=True, with_std=True, copy=True)

In [73]: China_test_days = Cmodel_neural.predict(test_China_scaled)
          print(China_test_days)
```

```
[3.10748125 2.12968554 5.28572345 3.30435985 ]
```

```
In [74]: US_test_days = Umodel_neural.predict(test_US_scaled)
          print(US_test_days)
```

```
[3.35775849 2.32990350 5.37432109 3.80004520 ]
```

Conclusions:

1. Based on the hypothesis test, datas from the unofficial datasets show that generally you can expect a better performance of Chinese package delivery industry.

2. Our ‘On Time’ probably prediction model can estimate the rate with an accuracy more than 70%.

3. Although the ‘Missed’ probability prediction model can obtain a higher accuracy, it seems that there’s still overfitting even using the classification tree. More precise datas are needed.

4. Generally, our model can predict the ‘Shipment Days’ with an error less than one day, which can be regarded as a good model cooperating with the ‘On Time’ probability model.

5. Actually, the origin dataset lacks of some important variables such as shipment distance. Maybe the models above can be improved with more precise datas.