به نام خدا

طبقه بندی دریافت سفارشات

زنجیره تامین کالاهای بهداشتی

"مدلسازی با Python"

پروژه درس داده کاوی

نام دانشجو: ثمین سلوکی

نام استاد: دکتر بزرگی امیری

نام کمک استاد: مهندس سارینا ملکی

پاییز ۱۴۰۲

فهرست مطالب

فهرست اشکال

مقدمه

تاریخ تحویل محموله های سفارشی در مدیریت زنجیره تامین، عامل مهمی در مدیریت تولید و موجودی، برنامه ریزی منابع و مدیریت ریسک است. تاخیر در تحویل، کاهش کارایی زنجیره ی تامین را به همراه خواهد داشت و می تواند موجب اختلال در زنجیره ی تامین شود. لذا انتخاب بهترین تامین کننده با در نظر گرفتن نوع کالا، حجم کالا، نحوه‌ی ارسال، بیمه کالا و... نقش کلیدی‌ای در مدیریت مخاطرات ایفا می‌کند. این پروژه از دیتاست مربوط به حمل و نقل کالاهای سلامت و قیمت گذاری زنجیره تامین(به طور خاص، مجموعه داده محموله های آزمایشگاهی ضد رتروویروسی به کشورهای حمایت شده)، برای یادگیری مدل طبقه بندی استفاده کرده است. نتیجه‌ی این پروژه مدیران را برای انتخاب بهترین تامین کننده یا انتخاب بهترین نوع سفارش اعم از حجم، نحوه ارسال، توافقات و غیره را یاری می‌کند.

فهم مسئله

آگاهی از این مسئله که آیا سفارش کالا با توجه به ارزش، نوع بیمه، حجم کالا، کشور مقصد و دیگر متغیرهای تاثیرگذار، تحویل دیرهنگام خواهد داشت یا خیر در انتخاب تامین کننده، مدیریت اختلالات، مدیریت انبار و موجودی و غیره تاثیر بسزایی دارد. رویکرد داده محور در این مسیله به کمک مدیران اجرایی خواهند آمد به که با تحلیل وضعیت و شرایط، انتخاب صحیح‌تر با قابلیت اطمینان بیشتری داشته باشند. در این پروژه، سعی شده است سفارشات کالاهای سلامت بر اساس تاریخ تحویل به دو گروه طبقه‌بندی شوند. گروه اول، شامل سفارشاتی هستند که تحویل زودهنگام و به‌هنگام داشته‌اند درحالی که گروه دوم، شامل سفارشاتی هستند که تحویل‌شان با تاخیر همراه بوده است.

درک داده

دیتاست اشاره شده از وب‌سایت catalog.data.gov دانلود شده‌است. این دیتاست متعلق به سازمان USAID [1] می‌باشد و شامل ۳۰ ستون و ۱۰۳۲۶ مشاهده است که در آن در طی ۹ سال، داده های مربوط به سفارشات محموله های بهداشتی ذخیره شده اند. توضیح متغیرهای مرتبط که در این پروژه استفاده شده اند به شرح زیر است:

- Country: کشور مشتری/سفارش دهنده (مقصد)
- managed by: مدیریت سفارش توسط دفتر مدیریت SCMS یا دفتر مدیریت برنامه (PMO) در ایالات متحده
- fulfill via: نحوه تامین سفارش از خرده فروش یا انبار شرکت.
- vendor inco term: نوع تعهدنامه درسفارشات که به توافقات مربوط به هزینه های بیمه، ارسال و ... اشاره دارد. (تعهدنامه سفارشات بین المللی).
- shipment mode: نحوه ارسال محموله از طریق مسیر های زمینی، آبی یا هوایی.
- scheduled delivery date: تاریخ برنامه ریزی شده ی دریافت محموله.
- delivered to client date: تاریخ دریافت محموله توسط مشتری.
- dosage form: نوع محصول (اعم از کپسول، محلول خوراکی، تزریقی).
- line item value : ارزش جمعی محموله. (به دلار آمریکا)
- unit price: هزینه هر قرص (برای داروها) یا هر آزمایش (برای کیت های آزمایش). (به دلار آمریکا)
- manufacturing site: محل تولید.
- line item insurance: هزینه‌ی بیمه. (به دلار آمریکا)
- weight: وزن محموله. (به کیلوگرم)
- freight cost: هزینه ترابری. (به دلار آمریکا)
- on time: کلاس تحویل به هنگام یا زودهنگام محموله و کلاس تحویل دیرهنگام محموله. (این متغیر از scheduled delivery date و delivered to client date ساخته شد و در دیتاست اصلی نبود.)

[1] US Agency for International Development

مسئله‌ای که در این دیتاست با ابهام همراه بود، مقادیری خاص در متغیر وزن بود. در بعضی مقادیر این متغیر اشاره می‌شد که نحوه‌ی پر کردنش از گزارش و شیوه‌ی خاصی پیروی می‌شده اما اطلاعات بیشتر در وبسایت مربوطه داده نشده‌بود و این مسئله مبهم باقی ماند و بالجبار مقادیر مربوطه ، null شدند. مسئله‌ی دیگر در کیفیت دادگان، در نظر گرفتن این است که در طی ۹ سال ثبت دیتا، نیروی انسانی مسئول دخیره کردن دیتا بوده‌اند لذا باید احتمال خطای انسانی را داد.

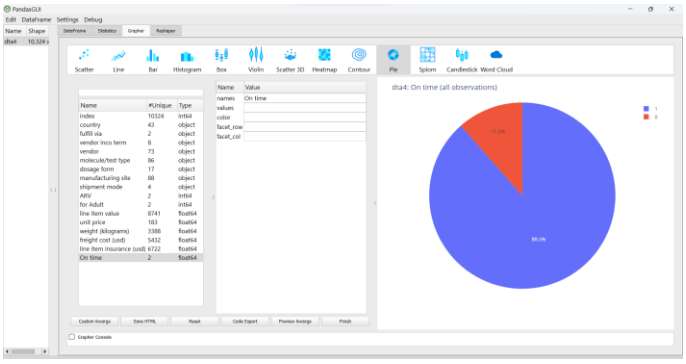
نوع داده‌ها به شرح زیر است:

shipment mode	delivered to client date	fulfill via	managed by	country
کیفی اسمی	تاریخ	کیفی اسمی	کیفی اسمی	کیفی اسمی
weight	scheduled delivery date	unit price	line item value	dosage form
کمی	تاریخ	کمی	کمی	کیفی اسمی
Ontime	line item insurance (usd)	manufacturing site	freight cost	vendor inco term
کیفی اسمی	کمی	کیفی اسمی	کمی	کیفی اسمی

شکل ۱ - نوع متغیر

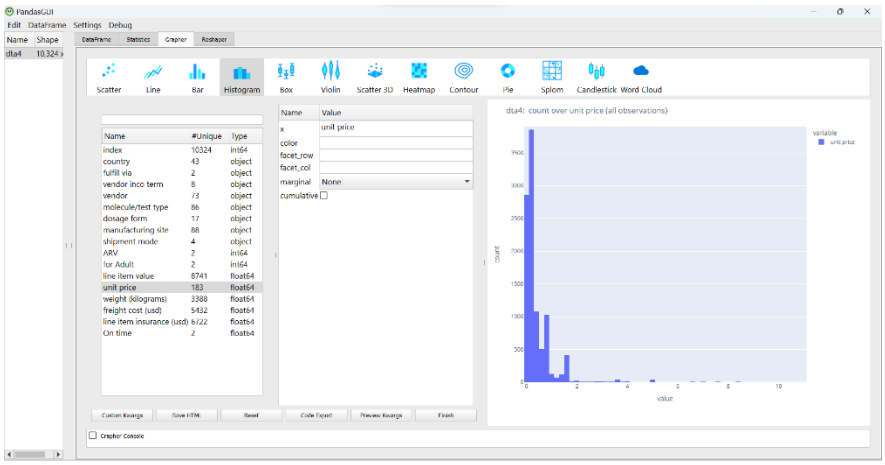
درک داده با PandasGUI

برای مرحله ی فهم داده ، تحلیل اکتشافی داده با کتابخانه ی pandasGUI انجام شد. در ادامه بخشی از عکس ها و تحلیل های انجام شده در این کتابخانه را مشاهده می‌کنید:

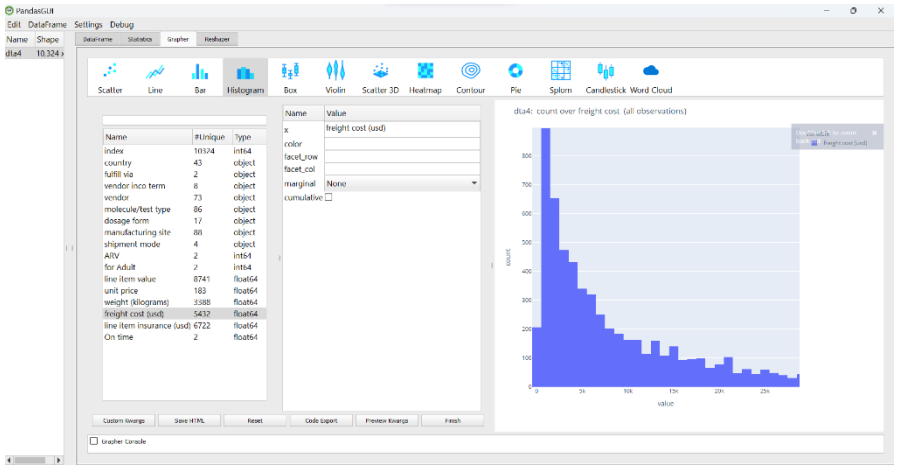


شکل ۲ - PandasGUI: OnTime

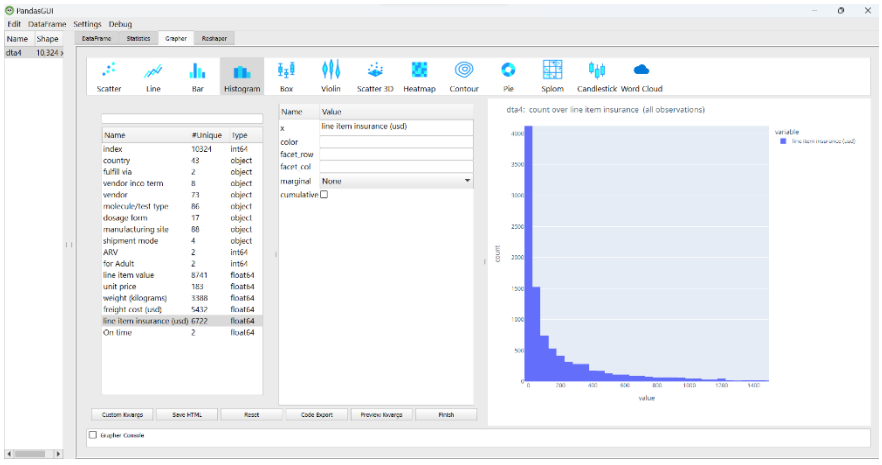
همان طور که از نمودار مشخص است، توزیع متغیر وابسته، به خوبی بین کلاس ها، بالانس نیست.



شکل ۳ - PandasGUI: hist Unit price

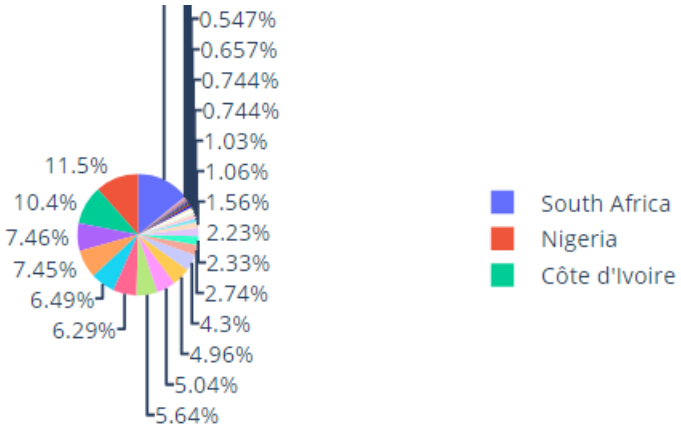


شکل ۴ - hist freight cost



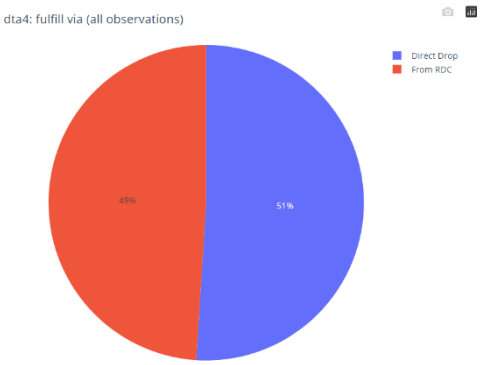
شکل ۵ - hist line item insurance

همان طور که از نمودارهای متغیر های وابسته مشخص است، توزیع هیچ کدام از متغیر ها نرمال نیست و نسبتا چولگی و کشیدگی بالایی دارند.

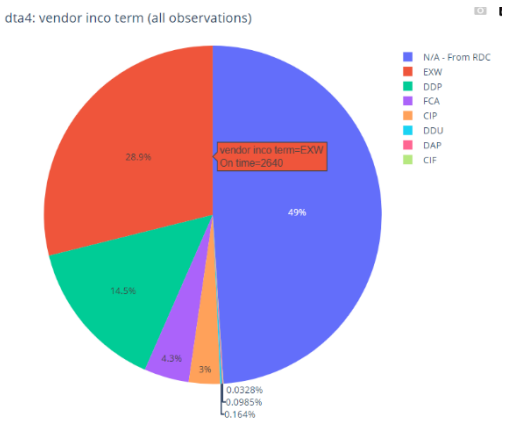


شکل ۶ - *Country VS Ontime*

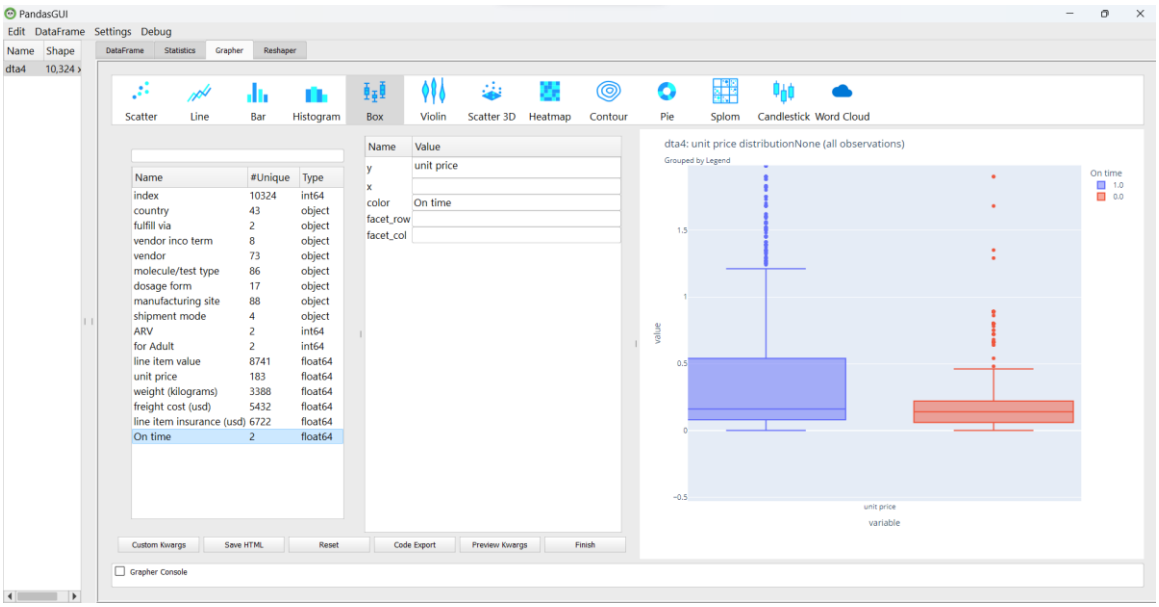
متغیر کیفی کشور مقصد سفارش دهنده، شامل ۴۳ کشور است. در ادامه خواهیم دید که کشور های با تعداد کمتر از ۴۰۰ سفارش، تجمیع شده و در گروه others قرار گرفتند.



شکل ۷ - *Filfill via VS Ontime*

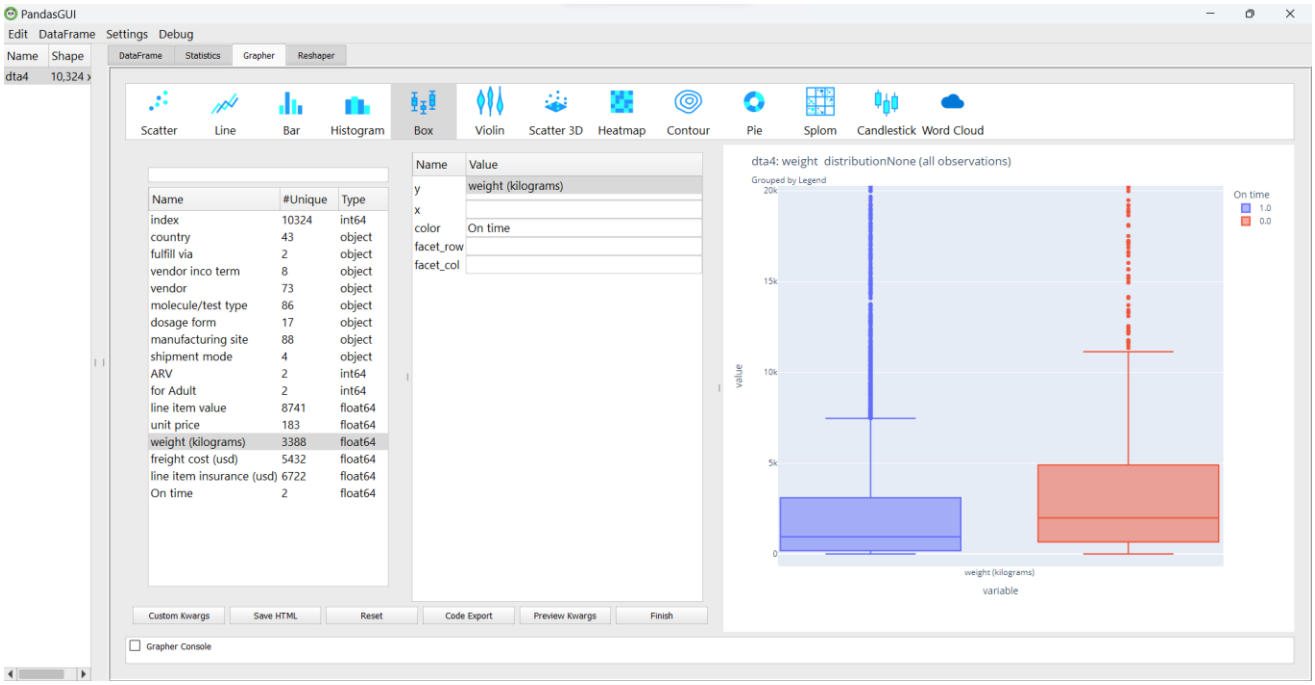


شکل ۸ - *Vendor inco term VS Ontime*



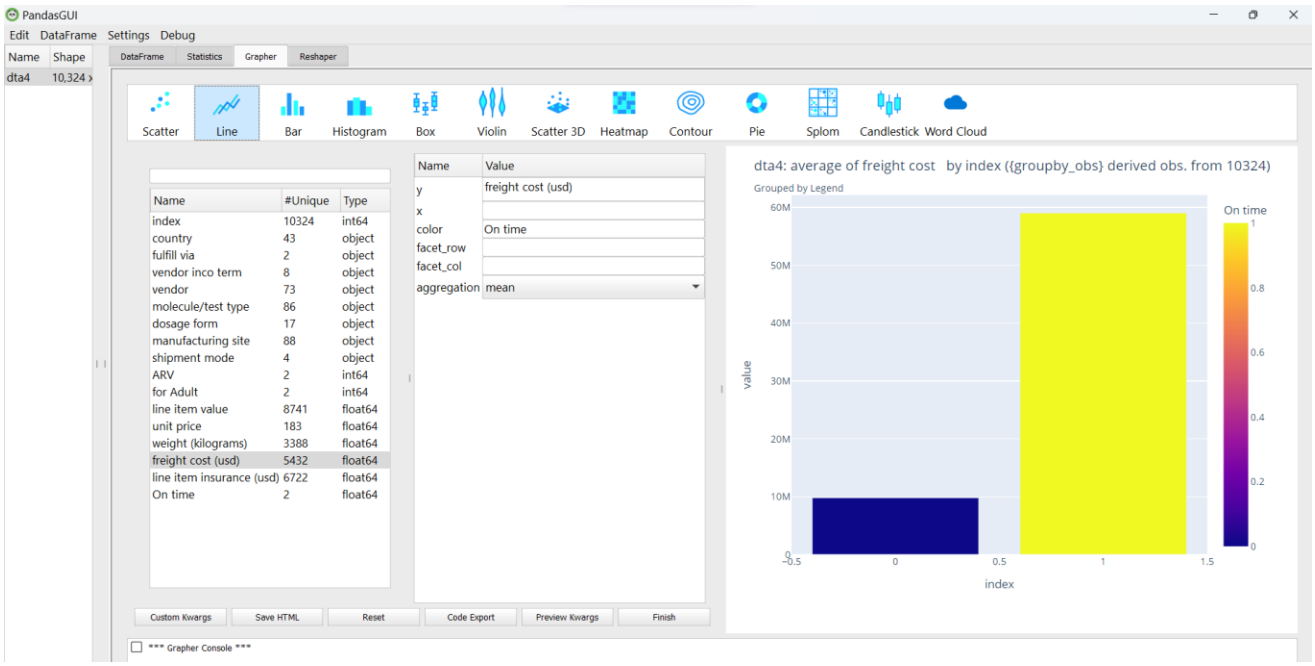
شکل ۹- *PandasGUI: Unit Price VS Ontime*

در نمودار مشاهده میکنیم که مقداری تفاوت بین کلاس ها در unit price مشاهده می‌شود. این تفاوت در چارک سوم و ماکسیمم توزیع ها، بارز است.



شکل ۱۰ - *PandasGUI: Weight VS Ontime*

در این توزیع، تفاوت محسوسی بین چارک سوم، میانه و ماکسیمم مقدار توزیع، مشخص است.



شکل ۱۱ - *Bar: Freight cost vs Ontime*

مشخصاً بین کلاس ها در متغیر freight cost تفاوت ظاهرا معنا داری وجود دارد.

آماده سازی داده

در آمیختن دیتا در این پروژه با توجه به متغیر وابسته و هدف پروژه نیاز نبود.

لازم به ذکر است که قدم به قدم تمییز کردن متغیر ها و ایجاد متغیر های جدید از دیتاست در فایل کد به جزییات در markdown گفته شده است.

پاک سازی داده

داده پرت

برای هر متغیر عددی، دادگانی که از بازه ی ۱.۵ برابر چارک اول و سوم متغیر خودشان خارج بودند به عنوان داده ی پرت در نظر گرفته شدند:

In [39]:
```
def nullify_outliers(data):
    Q1 = np.percentile((data).dropna(), 25)
    Q3 = np.percentile(data.dropna(), 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    nullified_data = np.where((data < lower_bound) | (data > upper_bound), np.nan, data)
    return nullified_data
dta15_cont=pd.DataFrame({
    'line_item_value' : pd.to_numeric(dta14.loc[:,'line item value'], errors='coerce'),
    'unit_price' : pd.to_numeric(dta14.loc[:, 'unit price'], errors='coerce'),
    'weight' : pd.to_numeric(dta14.loc[:,   'weight (kilograms)'], errors='coerce'),
    'freight_cost' : pd.to_numeric(dta14.loc[:, 'freight cost (usd)'], errors='coerce'),
    'line_item_insurance' : pd.to_numeric(dta14.loc[:, 'line item insurance (usd)'], errors='coerce')})
dta15_cont_cleaned=dta15_cont.copy()

for i in range(4):
    dta15_cont_cleaned.iloc[:,i]=nullify_outliers(dta15_cont.iloc[:,i])
dta15=pd.concat([dta14.iloc[:,[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  11, 15]],dta15_cont_cleaned],axis=1
```

شکل ۱۲- شناسایی و جایگزاری دادگان پرت

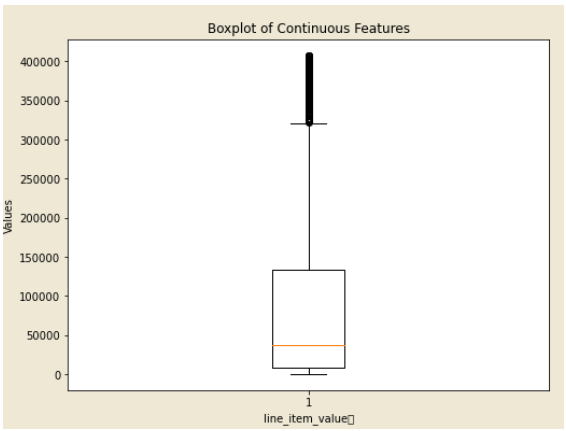
	line_item_value	unit_price	weight	freight_cost	line_item_insurance
count	1.032400e+04	10324.000000	6372.000000	6198.000000	10037.000000
mean	1.576506e+05	0.611701	3424.441306	11103.234819	240.117626
std	3.452921e+05	3.275808	13526.968270	15813.026692	500.190568
min	0.000000e+00	0.000000	0.000000	0.750000	0.000000
25%	4.314593e+03	0.080000	206.750000	2131.120000	6.510000
50%	3.047147e+04	0.160000	1047.000000	5869.655000	47.040000
75%	1.664471e+05	0.470000	3334.000000	14406.570000	252.400000
max	5.951990e+06	238.650000	857354.000000	289653.200000	7708.440000

```
dta15_cont_cleaned.describe()  #After
```

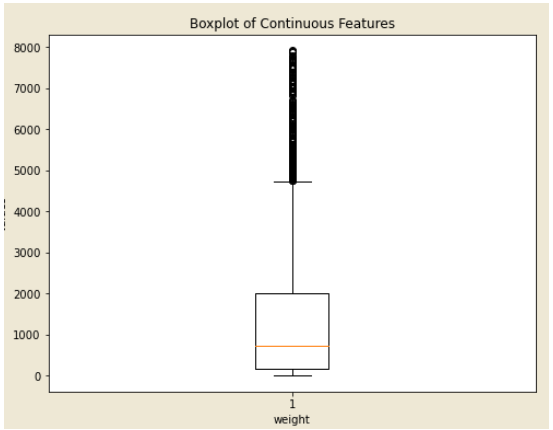
	line_item_value	unit_price	weight	freight_cost	line_item_insurance
count	9197.000000	9444.000000	5749.000000	5810.000000	10037.000000
mean	68444.601024	0.252289	1586.083319	8061.305084	240.117626
std	95163.119035	0.258073	1881.272511	7732.095373	500.190568
min	0.000000	0.000000	0.000000	0.750000	0.000000
25%	3290.000000	0.070000	169.000000	1945.105000	6.510000
50%	21000.000000	0.140000	807.000000	5221.335000	47.040000
75%	98784.000000	0.350000	2393.000000	12078.315000	252.400000
max	408815.920000	1.050000	8016.000000	32728.960000	7708.440000

شکل ۱۳ - گزارش قبل و بعد حذف داده های پرت

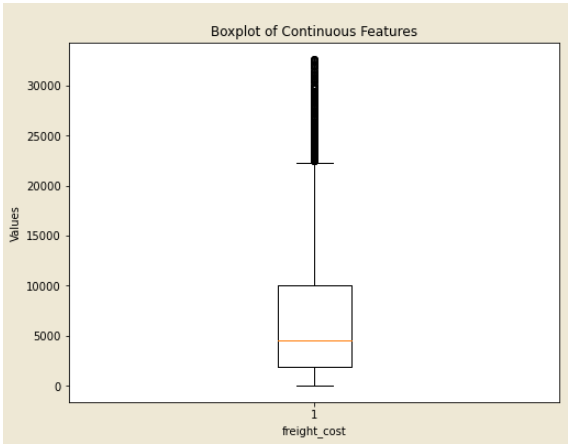
پس از حدف دادگان پرت، توزیع هر متغیر پیوسته به شکل زیر درآمد:



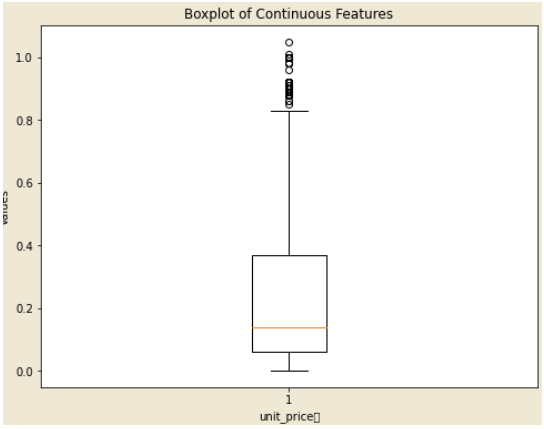
شکل ۱۵ - Matplotlib.pyplot: Line Item Value



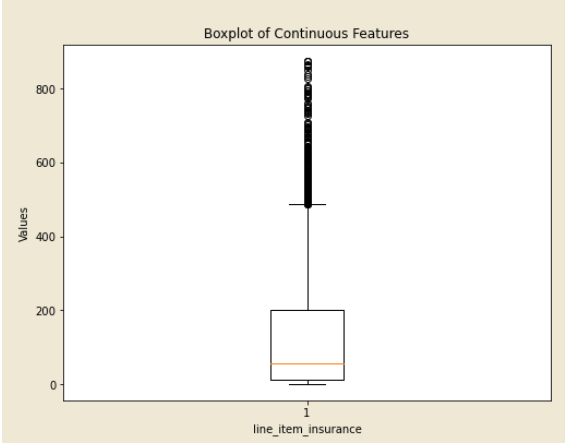
شکل ۱۴ - Matplotlib.pyplot: Weight



شکل ۱۷ - Matplotlib.pyplot:Freight Cost



شکل ۱۸ - - Matplotlib.pyplot:Unit Price



شکل ۱۶ - - Matplotlib.pyplot:Line Item Insurance

بعد از حدف داده‌های پرت متغیرهای کمی، تحلیل کواریانس انجام شد:

Step 20: *correlation Analysis for numeric variables*

```
corr_table1 = round( numerical_data.corr(method = 'pearson'), 2)
corr_table1
```

	line_item_value	unit_price	weight	freight_cost	line_item_insurance
line_item_value	1.00	0.14	0.32	0.30	0.15
unit_price	0.14	1.00	-0.08	0.09	0.13
weight	0.32	-0.08	1.00	0.43	0.27
freight_cost	0.30	0.09	0.43	1.00	0.26
line_item_insurance	0.15	0.13	0.27	0.26	1.00

شکل ۱۹- Correlation Table

بین متغیرهای کمی، کورلیشن نسبتاً کمی مشاهده می‌شود.

داده گمشده

برای جایگذاری دادگان گمشده ی متغیر ورن، از تابع simple imputer، میانگین وزن دادگان موجود جایگزاری شدند.

In [44]:
```
imputer = SimpleImputer(strategy='mean')
feature_15 = dta15.loc[:, 'line_item_insurance'].values.reshape(-1, 1)
dta15.loc[:, 'line_item_insurance'] = imputer.fit_transform(feature_15)
```

شکل ۲۰- Simple Imputer: mean

برای متغیر shipment method که از نوع کیفی اسمی است، نوع با بیشترین میزان وقوع برای جایگزاری انتخاب شد

shipment mode

```
In [45]: imputer = SimpleImputer(strategy= 'most_frequent')
         dta15.loc[:, 'shipment mode'] = imputer.fit_transform(dta15.loc[:, 'shipment mode'].values.reshape(-1,1
```

شکل ۲۱- *Simple Imputer : most frequent*

برای جایگزاری متغیر freight cost، از الگوریتم KNN استفاده شد:

freight cost (usd)

```
In [46]: from sklearn.impute import KNNImputer
         imputer = KNNImputer(n_neighbors=4, weights="uniform")
         dta15['freight_cost'] = imputer.fit_transform(dta15['freight_cost'].values.reshape(-1, 1))
```

شکل ۲۲- *KNNImputer*

برای جایگزاری دو متغیر line item value و unit price، با توجه به چولگی و کشیدگی قابل توجه این دو متغیر، به جای جایگزاری میانگین، میانه ی توزیع ها جایگزاری شد:

line_item_value

```
In [47]: from sklearn.impute import SimpleImputer
         imputer = SimpleImputer(strategy='median')
         feature_ = dta15.loc[:, 'line_item_value'].values.reshape(-1, 1)
         dta15.loc[:,'line_item_value'] = imputer.fit_transform(feature_)
```

شکل ۲۳- *Simple Imputer : median*

Unit price

```
In [48]: from sklearn.impute import SimpleImputer
         imputer = SimpleImputer(strategy='median')
         feature_ = dta15.loc[:, 'unit_price'].values.reshape(-1, 1)
         dta15.loc[:,'unit_price'] = imputer.fit_transform(feature_)
```

شکل ۲۴- *Simple Imputer : median*

تبدیل داده

در مرحله ی تبدیل داده، متغیر های کیفی به متغیر های کمی با روش onehotEncoder تبدیل شدند. همچنین برای نرمالسازی متغیر ها از تابع MinMaxScaler و StandardSclaer استفاده شد:

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder,MinMaxScaler
```

شکل ۲۵ - sklearn.preprocessing

```
    preprocessor = ColumnTransformer(
        transformers=[
            ('numerical_features', MinMaxScaler(), numerical_features),
            ('categorical_features', OneHotEncoder(sparse_output=False), categorical_features),
            ])
```

شکل ۲۶- MinMaxScaler – OneHotEncoder

در این مرحله، پس از تبدیل نوع متغیر به تاریخ، متغیر وابسته (هدف) هم ساخته شد:

```
dates_details=pd.DataFrame({
    'Delivered to client date' : dta4.iloc[:,6],
    'Scheduled delivery date': dta4.iloc[:,5],
    'dif':(dta4.iloc[:,6]-  dta4.iloc[:,5])})
from datetime import timedelta
dates_details_positive = dates_details[dates_details['dif'] > timedelta(days=0)]
dta13=dta12.copy()
dta13.loc[:,'On time']=np.nan
from datetime import timedelta
dates_details_positive = dates_details[dates_details['dif'] > timedelta(days=0)]
dta13.loc[ dates_details['dif'] <= timedelta(days=0)  ,'On time']=1
dta13.loc[ dates_details['dif'] > timedelta(days=0) ,'On time']=0
```

شکل ۲۷- timedelta- - Target Variable Definition

مدلسازی

پس از تقسیم داده ها به دو گروه آموزش و تست با نسبت ۷ به ۳، به مدلسازی می پردازیم:

Step 23: *deviding train/test*

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42,stratify=y)
```

شکل ۲۸- *train_test_split*

KNN

برای این مدلسازی از KNNClassifier از کتابخانه ی sklearn.neighbors استفاده شد:

In [267]:

```
numerical_features = [
    'line_item_value', 'unit_price', 'weight', 'freight_cost',
       'line_item_insurance'
]
categorical_features = [
    'country', 'fulfill via', 'vendor inco term', 'shipment mode', 'ARV',
       'for Adult', 'vendor', 'molecule/test type', 'dosage form',
       'manufacturing site'
                        ]

preprocessor = ColumnTransformer(
    transformers=[
        ('numerical_features', MinMaxScaler(), numerical_features),
        ('categorical_features', OneHotEncoder(sparse_output=False), categorical_features),
        ])
# ('other', 'passthrough', X.columns.difference(numerical_features).difference(categorical_features))
model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', KNeighborsClassifier())])
parameters = {
    'classifier__n_neighbors':[3, 4, 5, 6, 7, 8, 9, 10],
    'classifier__p': [1, 2, 3],
    'classifier__weights': ['uniform','distance']}

grid_search_KNN = GridSearchCV(model, parameters, cv=3,scoring='accuracy').fit(X_train, y_train)
```

شکل ۲۹ -*KNN*

بهترین متغیر های نتیجه شده از GridSearch، برای تعداد همسایه، ۹ عدد و P برای محاسبه فاصله، ۳ و وزن دهی یکنواخت اعلام شدند. دقت این مدل بر داده ها آموزش ، ۸۸/۱۱ درصد و بر داده های تست، ۸۷/۲۴ درصد می باشد.

SVM

برای این مدلسازی از SVC از کتابخانه ی sklearn.svm استفاده شد :

```
In [256]:
numerical_features = [
    'line_item_value', 'unit_price', 'weight', 'freight_cost',
      'line_item_insurance'
]
categorical_features = [
    'country', 'fulfill via', 'vendor inco term', 'shipment mode', 'ARV',
      'for Adult', 'vendor', 'molecule/test type', 'dosage form',
      'manufacturing site'
                        ]
preprocessor = ColumnTransformer(
    transformers=[
        ('numerical_features', StandardScaler(), numerical_features),
        ('categorical_features', OneHotEncoder(sparse_output=False), categorical_features),
        ])
svm_estimator_model = Pipeline([
    ('preprocessor', preprocessor),
    ('svc',SVC())  ])
parameters = {'svc__C': [0.001, 0.01, 0.1, 1, 10],'svc__gamma': [0.001, 0.01, 0.1, 1, 10]}

In [257]:
from sklearn.model_selection import GridSearchCV
grid_cv_SVM = GridSearchCV(svm_estimator_model, param_grid=parameters, cv=5)
grid_cv_SVM.fit(X_train, y_train)
```

شکل ۳۰- SVM

بهترین متغیر های نتیجه شده از GridSearch، برای C، ۱ و برای گاما ۰.۱ اعلام شدند. دقت این مدل بر داده ها آموزش ، ۸۸/۶۲ درصد و بر داده های تست، ۸۷/۳۴ درصد می باشد.

DT

برای این مدلسازی از DecisionTreeClassifier از کتابخانه ی sklearn.tree استفاده شد.

```
In [293]:
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

In [294]:
numerical_f =  [
    'line_item_value', 'unit_price', 'weight', 'freight_cost',
       'line_item_insurance'
]
categorical_f =  [
    'country', 'fulfill via', 'vendor inco term', 'shipment mode', 'ARV',
       'for Adult', 'vendor', 'molecule/test type', 'dosage form',
       'manufacturing site']
preprocessor = ColumnTransformer(
                transformers=[
                ('for numerical_f',MinMaxScaler(), numerical_f ),
                ('for categorical_f',OneHotEncoder(), categorical_f)
                ])
dt_model = Pipeline([
    ('preprocessor', preprocessor),
    ('dt',DecisionTreeClassifier())  ])
parameters = {
    'dt__criterion': ['gini', 'entropy'],
    'dt__max_depth': [5, 10, 15, 20],
    'dt__min_samples_leaf': [5, 10, 20]  }
grid_DT= GridSearchCV(dt_model, param_grid=parameters, cv=5)
grid_DT.fit(X_train, y_train)
```

شکل ۳۱- Decision Tree

بهترین متغیر های نتیجه شده از GridSearch، برای شاخص تقسیم، Gini و برای بیشینه عمق درخت، ۵ و برای کمترین تعداد نمونه در هر برگ ،۲۰ اعلام شدند. دقت این مدل بر داده ها آموزش ، ۸۸/۴۷ درصد و بر داده های تست، ۸۷/۸۳ درصد می باشد.

RandomForest

برای این مدلسازی از RandomForestClassifier از کتابخانه ی sklearn.ensemble استفاده شد.

```
from sklearn.ensemble import RandomForestClassifier

numerical_features = ['line_item_value', 'unit_price', 'weight', 'freight_cost', 'line_item_insurance']
categorical_features = ['country', 'fulfill via', 'vendor inco term', 'shipment mode', 'ARV',
                        'for Adult', 'vendor', 'molecule/test type', 'dosage form', 'manufacturing sit
preprocessor = ColumnTransformer(
    transformers=[
        ('numerical_features', StandardScaler(), numerical_features),
        ('categorical_features', OneHotEncoder(sparse=False), categorical_features),])
rf_model = Pipeline([
    ('preprocessor', preprocessor),
    ('rf', RandomForestClassifier(random_state=42))
])
rf_model.fit(X_train, y_train)
```

شکل ۳۲- Random Forest

بهترین مقادیر مربوطه با توجه به نتیجه ی GridSearch، برابرند با: بیشینه عمق درخت: ۱۰ و کمینه تعداد نمونه در هر برگ:۲، کمینه تعداد بخش، ۱۰ و تعداد درختان جنگل ۲۰۰ بودند. دقت این مدل بر روی داده های آموزش ۹۹/۸۲ و بر داده های تست، ۸۸/۱۲ بود. بدلیل خطر overfitting، seed اولیه تقسیم داده های تست و آموزش تغییر داده شدند و مجدد دقت مدل سنجیده شد؛ دفعه ی ثانویه، دقت مدل آموزش ۹۹/۸۳ بود.

Naïve Bayes

برای این مدلسازی از GaussianNB از کتابخانه ی sklearn.naive_bayes استفاده شد.

```
from sklearn.metrics import accuracy_score

from sklearn.naive_bayes import GaussianNB

numerical_features = ['line_item_value', 'unit_price', 'weight', 'freight_cost', 'line_item_insurance']
categorical_features = ['country', 'fulfill via', 'vendor inco term', 'shipment mode', 'ARV',
                        'for Adult', 'vendor', 'molecule/test type', 'dosage form', 'manufacturing sit
preprocessor = ColumnTransformer(
    transformers=[
        ('numerical_features', StandardScaler(), numerical_features),
        ('categorical_features', OneHotEncoder(sparse=False), categorical_features),
    ])
nb_model = Pipeline([     ('preprocessor', preprocessor),
    ('nb', GaussianNB())])
nb_model.fit(X_train, y_train)
```

شکل ۳۳- Naive Bayes

این مدل ضعیف ترین عملکرد را بین مدل ها داشت! دقت بر داده های آموزش، ۳۹/۳ درصد و بر داده های تست، ۳۷ درصد بود! بدلیل مقدار پایین دقت، مجدد نمونه گیری با یک عدد تصادفی seed دیگر انتخاب شد که مجدد دقت پایینی(۳۷/۴۸ درصد) داشت.

ارزیابی

در این مرحله، با استفاده از توابع زیر، تک تک مدل‌ها ارزیابی شدند:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

شکل34- sklearn metrics

در جدول شماره ی ۳۴، گزارشی از عملکرد مدل ها تحت شاخص های متفاوت ارایه شده است. بهترین مقدار شاخص در هر ستون، با رنگ قرمز مشخص شده است و بهترین دومین مقدار با رنگ سبز مشخص شده است.

	Train	Test			
Model	Accuracy	Accuracy	Precision	Recall	F1 Score
KNN	0.881123792	0.872498386	0.826316306	0.872498386	0.840982268
SVM	0.886244071	0.883473209	0.819456078	0.883473209	0.832744704
DT	0.884721772	0.878308586	0.826513172	0.839317331	0.839317331
RF	0.998200941	0.883795997	0.822435286	0.883795997	0.832912751
NB	0.393025187	0.370561653	0.891579823	0.370561653	0.429557999

شکل35 -Model Evaluation

با توجه به اینکه هرکدام از این معیار ها معنای متفاوتی دارند، بنابر اولویت و نظر مدیریت، مدل برتر انتخاب و استفاده می شود.

نتیجه‌گیری

تجربه، نتیجه و یادگیری از داده کاوی

بزرگترین چالش و یادگیری‌ای که من در این پروژه روبرو بودم، نحوه‌ی برخورد با دادگان کیفی و encode کردن آنها بود. پس از استفاده از pipeline از کتابخانه ی sklearn توانستم به این چالش غلبه کنم.

چالش دیگری که با آن روبرو شدم، نمایش دادگان بود. این امکان وجود داشت که از کتابخانه ی matplotlib استفاده کنم اما تصمیم گرفتم از کتابخانه ی PandasGUI برای فهم داده و تحلیل اکتشافی داده استفاده کنم. در این کتابخانه این امکان فراهم بود که مانند نرم افزارPower BI نمودار کشیده شود و همچنین تحلیل های آماری ای هم از داده ها ارایه می کرد. لذا امکانات کد های matplotlib و دیگر کد هایی چون value_counts یا describe در این کتابخانه خلاصه شده بود و بدلیل رابط کاربری خوبی که داشت، به matplotlib ترجیح داده شد. یک قابلیت جالب دیگر این کتابخانه، توانایی تغییر داده ها مانند نرم افزار اکسل فراهم بود.

نتیجه و ارزش خلق‌شده از پروژه‌ی طبقه بندی سفارشات

در این پروژه تلاش شد که سفارشات با در نظر گرفتن متغیرهای کشور مقصد، دفتر مدیریت سفارش، نوع تامین‌کننده‌ی سفارش، نوع تعهدنامه‌ی بین المللی خرید کالا، نحوه ارسال محموله، تاریخ برنامه ریزی شده و تاریخ دریافت محموله، نوع محصول، ارزش جمعی محصول، هزینه هر عدد محصول، محل تولید، وزن محموله و هزینه‌ی ترابری، سعی بر طبقه‌بندی سفارشات و تامین کننده شد. نتیجه ی این پروژه برای مدیران عالی و مدیران اجرایی کاربرد دارد بدین صورت که ایشان را در انتخاب و تنظیم بهترین سفارش، اعم از انتخاب مقدار سفارش، انتخاب نوع تعهدنامه بین الملل، انتخاب نحوه ی ارسال، انتخاب خرید از خرده فروش یا تولیدکننده و غیره با تمرکز بر موقع رسیدن یا تعدی از مهلت سفارش، یاری می کند. با استفاده از این تصمیم داده محور، می‌توان خط تولید با پایداری بیشتری را طراحی کرد و زنجیره‌ی تامین مقاوم تری را طراحی کرد.