In [1]:
```
#Analytics Vidya Hackaton-big mart sales
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
import io
%cd "C:\Users\uttej\OneDrive\Desktop\Python"
```

```
C:\Users\uttej\OneDrive\Desktop\Python
```

In [4]:
```
bigmarttrain=pd.read_csv("train_v9rqX0R.csv")
```

In [5]:
```
bigmarttest=pd.read_csv("test_AbJTz2l.csv")
```

In [6]:
```
bigmarttrain.head()
```

Out[6]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Iden
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OU
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OU
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OU
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OU
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OU

In [7]:
```
bigmarttrain.tail()
```

Out[7]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_I
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	

In [8]:
```
bigmarttrain.info
```

Out[8]:
```
<bound method DataFrame.info of      Item_Identifier  Item_Weight Item_Fat_Content
Item_Visibility  \
0              FDA15        9.300          Low Fat         0.016047
1              DRC01        5.920          Regular         0.019278
2              FDN15       17.500          Low Fat         0.016760
3              FDX07       19.200          Regular         0.000000
4              NCD19        8.930          Low Fat         0.000000
...              ...          ...              ...              ...
8518           FDF22        6.865          Low Fat         0.056783
8519           FDS36        8.380          Regular         0.046982
8520           NCJ29       10.600          Low Fat         0.035186
8521           FDN46        7.210          Regular         0.145221
8522           DRG01       14.800          Low Fat         0.044878

                  Item_Type  Item_MRP Outlet_Identifier  \
0                     Dairy  249.8092            OUT049
1               Soft Drinks   48.2692            OUT018
2                      Meat  141.6180            OUT049
3     Fruits and Vegetables  182.0950            OUT010
4                 Household   53.8614            OUT013
...                     ...       ...               ...
8518            Snack Foods  214.5218            OUT013
8519          Baking Goods  108.1570            OUT045
8520    Health and Hygiene   85.1224            OUT035
8521            Snack Foods  103.1332            OUT018
8522            Soft Drinks   75.4670            OUT046

      Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                          1999      Medium               Tier 1
1                          2009      Medium               Tier 3
2                          1999      Medium               Tier 1
3                          1998         NaN               Tier 3
4                          1987        High               Tier 3
...                         ...         ...                  ...
8518                       1987        High               Tier 3
8519                       2002         NaN               Tier 2
8520                       2004       Small               Tier 2
8521                       2009      Medium               Tier 3
8522                       1997       Small               Tier 1

           Outlet_Type  Item_Outlet_Sales
0     Supermarket Type1          3735.1380
1     Supermarket Type2           443.4228
2     Supermarket Type1          2097.2700
3         Grocery Store           732.3800
4     Supermarket Type1           994.7052
...                 ...                ...
8518  Supermarket Type1          2778.3834
8519  Supermarket Type1           549.2850
8520  Supermarket Type1          1193.1136
8521  Supermarket Type2          1845.5976
8522  Supermarket Type1           765.6700

[8523 rows x 12 columns]>
```

In [9]:
```
bigmarttrain.dtypes
```

Out[9]:
```
Item_Identifier               object
Item_Weight                  float64
Item_Fat_Content              object
```

```
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
Outlet_Size                   object
Outlet_Location_Type          object
Outlet_Type                   object
Item_Outlet_Sales            float64
dtype: object
```

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

Out[10]:
```
(8523, 12)
```

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

Out[11]:
```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility',
       'Item_Type', 'Item_MRP', 'Outlet_Identifier',
       'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Location_Type',
       'Outlet_Type', 'Item_Outlet_Sales'],
      dtype='object')
```

In [12]:
```
# Data preprocessing or feature engineering
# 1) Check for duplicate variables might have same name or different name.
# Inspection of variables.Same min,max,median,Q1,Q2,
# Write for a loop

# 2)Single value variables.whole column has same value like 0 or 1
# Y ,etc.
# 3) Missing value imputation-By default python will impute null cells
# with NaN or NA
# a)Drop the missing value rows.if NA are less 1% and large data available
# b)If the variable has <70% missing values then impute the word
# "missing " or "not available"
# c)if less then 70% missing values then manual imputation can be done
# numeric variable-mean or median ; categorial-mode or most frequent
# manual imputation does not take into consideration impact for other imdependent va
# d)algorithm based missing value imputation
# 1)Multivariate imputation using chained equations(MICE)-impute simultaneously both
# and non numeric variables
# 2)KNN imputer-K nearest neighbors algorithm is used for missing value imputation
# KNN uses euclidean distance and imputes with closest distance
# 3)Probablistic PCA-Principal component analysis algoritm is used for imputation

# 4)Data transformation-transformation of dependent and independent variables
# a)dependent Variable:
# positive skewness-logarmathic transformation or sqaure root transformation
# negative skewness-exponantial transformation or power transformation
# independent variables-transformation needed if they are in different mathematical
# 1)Standard scalar-Z score=(x-mean)/standard deviation.standard scalar will give yo
# 2)Min-max scalar -(X-Xmin)/(Xmax-Xmin) min max scalar will scale all observations
# 3)Correction of duplicate levels in categorial variables
# Gender m,M,male,MALE
# 4)Dummy variable encoding-assigning numeric identifiers to the classes or levels o
# a)Lable Encoding-Encode in the same and overwrite column
# Gender-M,F,F,F,M,M,M,M
# Gender-1,0,0,0,1,1,1,1
# b)pd.get_dummies-Create new columns based on number of classes or levels and does
# Gender-M,F,F,F,M,M,M,M
# Gender_M-1,0,0,0,1,1,1,1
```

```
# Gender_F-0,1,1,1,0,0,0,0
# 5)Create new variables from existing variables
# Dealing with dates.Dates by default are read as objects and dates must be converte
# 6)Extracting information from date like year,month,day of week,quarter,ismonthstar
# 7)Dimensionality reduction or reducing the number of variables.
```

In [13]:
```
bigmarttrain.isnull().sum().sort_values(ascending=False)#Identify NA
```

Out[13]:
```
Outlet_Size                2410
Item_Weight                1463
Item_Identifier               0
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

In [14]:
```
bigmarttest.isnull().sum().sort_values(ascending=False)
```

Out[14]:
```
Outlet_Size                1606
Item_Weight                 976
Item_Identifier               0
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Location_Type          0
Outlet_Type                   0
dtype: int64
```

In [15]:
```
print(bigmarttrain.shape)
print(bigmarttest.shape)
```

```
(8523, 12)
(5681, 11)
```

In [16]:
```
#add dependent variable to test data for concatenation of data frames
bigmarttest ["Item_outlet_Sales"]="test"
```

In [20]:
```
#row wise concatenation-paste all rows of train data first and below paste all rows
combinedf=pd.concat([bigmarttrain,bigmarttest],axis=0)
#axis=0 is row concatentation
#axis=1 is column concatentation
```

In [21]:
```
combinedf.shape
```

Out[21]:
```
(14204, 13)
```

In [24]:
```
combinedf.Outlet_Size.value_counts(dropna=False)
```

Out[24]:
```
Medium    4655
NaN       4016
Small     3980
High      1553
Name: Outlet_Size, dtype: int64
```

combinedf.Outlet_Size=combinedf.Outlet_Size.fillna("Missing")

In [25]:
```
combinedf.Outlet_Size=combinedf.Outlet_Size.fillna("Missing")
```

In [27]:
```
combinedf.Item_Weight.describe()
```

Out[27]:
```
count    11765.000000
mean        12.792854
std          4.652502
min          4.555000
25%          8.710000
50%         12.600000
75%         16.750000
max         21.350000
Name: Item_Weight, dtype: float64
```

In [29]:
```
combinedf.Item_Weight=combinedf.Item_Weight.fillna(
combinedf.Item_Weight.mean()) # mean Imputation
```

In [30]:
```
combinedf.columns
```

Out[30]:
```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility',
       'Item_Type', 'Item_MRP', 'Outlet_Identifier',
       'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Location_Type',
       'Outlet_Type', 'Item_Outlet_Sales', 'Item_outlet_Sales'],
      dtype='object')
```

In [31]:
```
combinedf.Item_Identifier.head()
```

Out[31]:
```
0    FDA15
1    DRC01
2    FDN15
3    FDX07
4    NCD19
Name: Item_Identifier, dtype: object
```

In [32]:
```
combinedf['Item_Code']=combinedf.Item_Identifier.apply(
lambda x:x[0:2])
```

In [35]:
```
combinedf.Item_Code.value_counts()
```

Out[35]:
```
FD    10201
NC     2686
DR     1317
Name: Item_Code, dtype: int64
```

In [36]:
```
combinedf.Item_Fat_Content.value_counts()
```

Out[36]:
```
Low Fat    8485
Regular    4824
LF          522
```

```
reg         195
low fat     178
Name: Item_Fat_Content, dtype: int64
```

In [37]:
```
combinedf.Item_Fat_Content=combinedf.Item_Fat_Content.replace(
['LF','low fat'],'Low Fat')
```

In [38]:
```
combinedf.Item_Fat_Content=combinedf.Item_Fat_Content.replace(
'reg','Regular')
```

In [39]:
```
combinedf.Item_Visibility.describe()
```

Out[39]:
```
count    14204.000000
mean         0.065953
std          0.051459
min          0.000000
25%          0.027036
50%          0.054021
75%          0.094037
max          0.328391
Name: Item_Visibility, dtype: float64
```

In [40]:
```
combinedf.Item_MRP.describe()
```

Out[40]:
```
count    14204.000000
mean       141.004977
std         62.086938
min         31.290000
25%         94.012000
50%        142.247000
75%        185.855600
max        266.888400
Name: Item_MRP, dtype: float64
```

In [41]:
```
combinedf.Outlet_Identifier.value_counts()
```

Out[41]:
```
OUT027    1559
OUT013    1553
OUT049    1550
OUT046    1550
OUT035    1550
OUT045    1548
OUT018    1546
OUT017    1543
OUT010     925
OUT019     880
Name: Outlet_Identifier, dtype: int64
```

In [42]:
```
# Calculate Outlet_Age from Outlet_Establishment_Year as of 2021
combinedf['Outlet_Age']=2021-combinedf.Outlet_Establishment_Year
```

In [43]:
```
combinedf.Outlet_Age.describe()
```

Out[43]:
```
count    14204.000000
mean        23.169319
std          8.371664
min         12.000000
```

```
25%         17.000000
50%         22.000000
75%         34.000000
max         36.000000
Name: Outlet_Age, dtype: float64
```

In [44]:
```
combinedf.Outlet_Location_Type.value_counts()
```

Out[44]:
```
Tier 3    5583
Tier 2    4641
Tier 1    3980
Name: Outlet_Location_Type, dtype: int64
```

In [45]:
```
combinedf.Outlet_Type.value_counts()
```

Out[45]:
```
Supermarket Type1    9294
Grocery Store        1805
Supermarket Type3    1559
Supermarket Type2    1546
Name: Outlet_Type, dtype: int64
```

In [48]:
```
combinedf.dtypes
```

Out[48]:
```
Item_Identifier               object
Item_Weight                  float64
Item_Fat_Content              object
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
Outlet_Size                   object
Outlet_Location_Type          object
Outlet_Type                   object
Item_Outlet_Sales            float64
Item_outlet_Sales             object
Item_Code                     object
Outlet_Age                     int64
dtype: object
```

In [49]:
```
combinedf=combinedf.drop(['Item_Identifier',
                          'Outlet_Establishment_Year'],axis=1)
```

In [50]:
```
# Split Data into nuemric & object data
numcols=combinedf.select_dtypes(include=np.number)
objectcols=combinedf.select_dtypes(include=['object'])
```

In [57]:
```
print(numcols.shape)
print(objectcols.shape)
```

```
(14204, 5)
(14204, 8)
```

In [62]:
```
# Move Item_Outlet_Sales from object to Numeric. Since it contains
# text values 'test' it is showing as object.
numcols["Item_Outlet_Sales"]=objectcols.Item_Outlet_Sales()
```

```
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_564/2768168014.py in <module>
      1 # Move Item_Outlet_Sales from object to Numeric. Since it contains
      2 # text values 'test' it is showing as object.
----> 3 numcols["Item_Outlet_Sales"]=objectcols.Item_Outlet_Sales()

~\anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5485         ):
   5486             return self[name]
-> 5487         return object.__getattribute__(self, name)
   5488
   5489     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'Item_Outlet_Sales'
```

In [67]:

```
objectcols=objectcols.drop("Item_Outlet_Sales",axis=1)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_564/703209927.py in <module>
----> 1 objectcols=objectcols.drop("Item_Outlet_Sales",axis=1)

~\anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

~\anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, inde
x, columns, level, inplace, errors)
   4904                 weight  1.0     0.8
   4905         """
-> 4906         return super().drop(
   4907             labels=labels,
   4908             axis=axis,

~\anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, ind
ex, columns, level, inplace, errors)
   4148         for axis, labels in axes.items():
   4149             if labels is not None:
-> 4150                 obj = obj._drop_axis(labels, axis, level=level, errors=error
s)
   4151
   4152         if inplace:

~\anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axi
s, level, errors)
   4183                 new_axis = axis.drop(labels, level=level, errors=errors)
   4184             else:
-> 4185                 new_axis = axis.drop(labels, errors=errors)
   4186             result = self.reindex(**{axis_name: new_axis})
   4187

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, erro
rs)
   6015         if mask.any():
   6016             if errors != "ignore":
-> 6017                 raise KeyError(f"{labels[mask]} not found in axis")
   6018             indexer = indexer[~mask]
   6019         return self.delete(indexer)

KeyError: "['Item_Outlet_Sales'] not found in axis"
```

In [64]:
```
from sklearn.preprocessing import LabelEncoder
```

In [65]:
```
le=LabelEncoder()
```

In [66]:
```
objectcolsencode=objectcols.apply(le.fit_transform)
```

In [68]:
```
objectcols.head(2)
```

Out[68]:

	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	It
0	Low Fat	Dairy	OUT049	Medium	Tier 1	Supermarket Type1	
1	Regular	Soft Drinks	OUT018	Medium	Tier 3	Supermarket Type2	

In [69]:
```
objectcolsencode.head(2)
```

Out[69]:

	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	It
0	0	4	9	1	0	1	
1	1	14	3	1	2	2	

In [70]:
```
objectcolsdummy=pd.get_dummies(objectcols)
```

In [71]:
```
objectcolsdummy.head()
```

Out[71]:

	Item_Fat_Content_Low Fat	Item_Fat_Content_Regular	Item_Type_Baking Goods	Item_Type_Breads	Item_Type_
0	1	0	0	0	
1	0	1	0	0	
2	1	0	0	0	
3	0	1	0	0	
4	1	0	0	0	

5 rows × 43 columns

In [72]:
```
# Columnwise Concatenation
combinedf_clean=pd.concat([numcols,objectcolsencode],axis=1)
```

In [73]:
```
bigmarttraindf=combinedf_clean[
    combinedf_clean.Item_Outlet_Sales!='test']
```

```
bigmarttestdf=combinedf_clean[
    combinedf_clean.Item_Outlet_Sales=='test']
```

In [74]:
```
bigmarttestdf=bigmarttestdf.drop('Item_Outlet_Sales',axis=1)
```

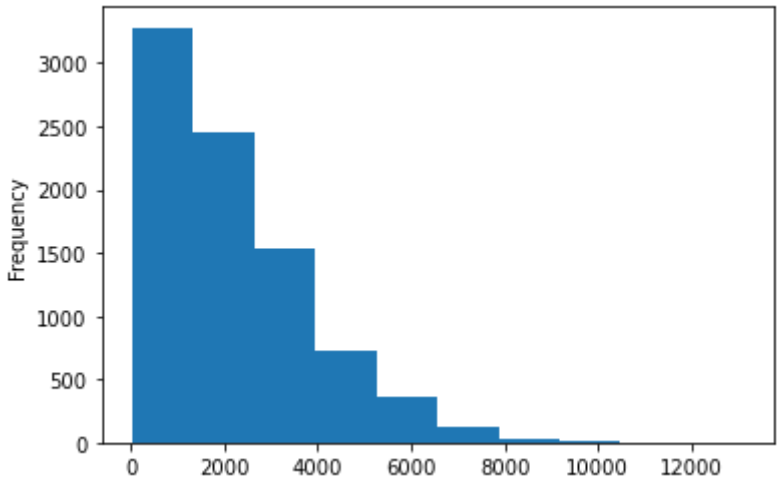
In [75]:
```
bigmarttraindf.Item_Outlet_Sales.dtypes
```

Out[75]:
```
dtype('float64')
```

In [76]:
```
bigmarttraindf.Item_Outlet_Sales=bigmarttraindf.Item_Outlet_Sales.astype('float64')
```

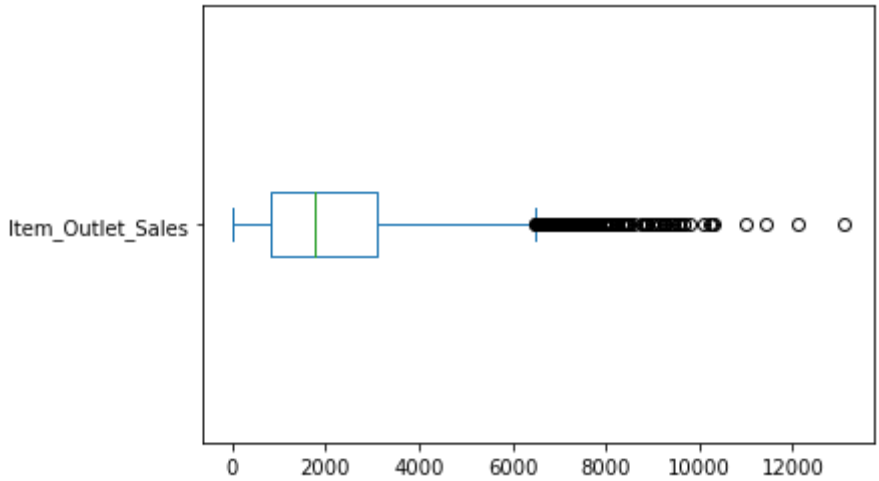
In [77]:
```
# Create Histogram, Boxplot and Density Curve of Item_Outlet_Sales
bigmarttraindf.Item_Outlet_Sales.plot(kind='hist')
# Positive Skewness & Kurtosis
print("Skew:",bigmarttraindf.Item_Outlet_Sales.skew())
print("Kurt:",bigmarttraindf.Item_Outlet_Sales.kurt())
```

```
Skew: 1.1775306028542796
Kurt: 1.6158766814287264
```



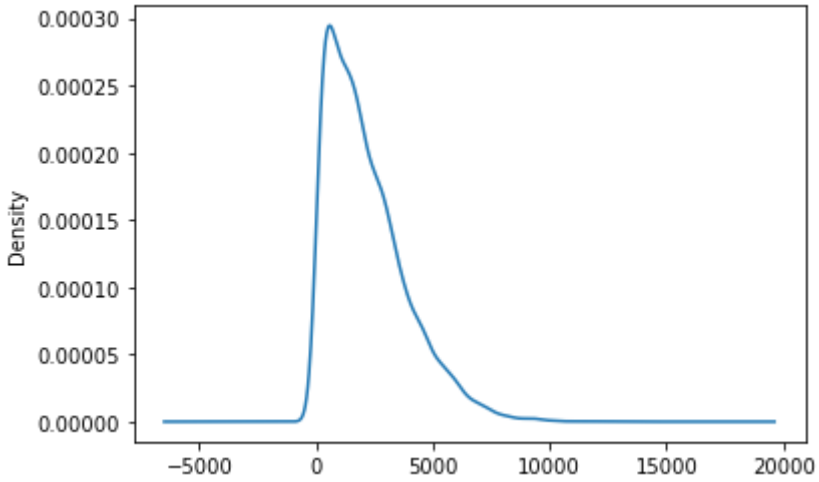
In [78]:
```
bigmarttraindf.Item_Outlet_Sales.plot(kind='box',vert=False)
```

Out[78]:
```
<AxesSubplot:>
```



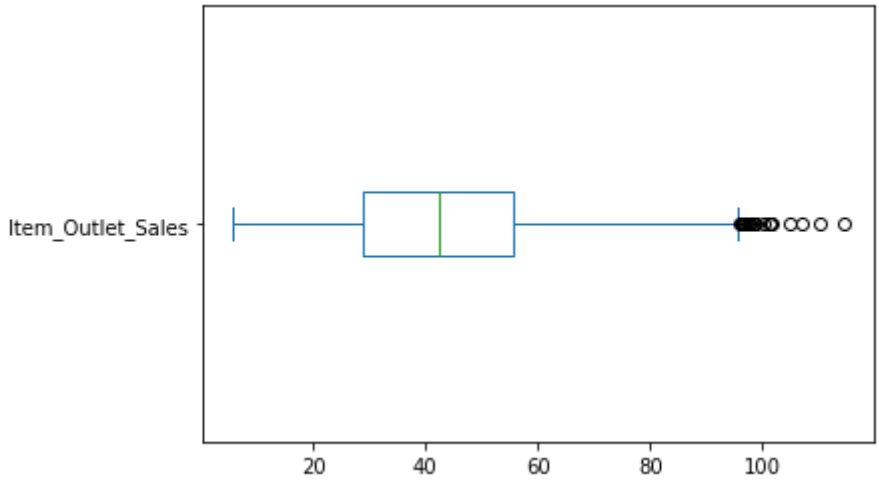
In [79]:
```
bigmarttraindf.Item_Outlet_Sales.plot(kind='density')
```

Out[79]:
```
<AxesSubplot:ylabel='Density'>
```



In [80]:
```
np.sqrt(bigmarttraindf.Item_Outlet_Sales).plot(kind='box',vert=False)
```

Out[80]:
```
<AxesSubplot:>
```



In [81]:
```
# Inferential Statistics - Hypothesis testing
# Hypothesis testing is comparision of means or averages of or more
# than 2 classes or levels in categorical var with a numeric variable

# Relationship between 2 Numeric Variables - Covariance and Correlation
# Relationship between One numeric variable and other categorical
# variable - Hypothesis testing
# Relationship between 2 Non numeric categorical variables.

# Fundamental Assumtion of Hypothesis testing is means or averages
# of classes or levels must be different. groupby()
# Assumption of Numeric Variable must be numerical continuos,closer
# to normal distrinution with minimum outliers and no missing values.

# One Numeric variable and other variable is categrocial with
# exactly 2 levels - 2 Sample Independent t test
```

In [82]:
```
# What is the Average Item_Outlet_Sales of different Item_Fat_Content?
bigmarttraindf.Item_Outlet_Sales.groupby(
    bigmarttraindf.Item_Fat_Content).mean()
# 0 - Low fat & 1 - Regular
```

Out[82]:
```
Item_Fat_Content
0    2157.711534
1    2224.561170
Name: Item_Outlet_Sales, dtype: float64
```

In [83]:
```
bigmarttraindf.Item_Outlet_Sales.groupby(
    bigmarttraindf.Item_Fat_Content).var() # Unequal Variance
```

Out[83]:
```
Item_Fat_Content
0    2.883115e+06
1    2.963496e+06
Name: Item_Outlet_Sales, dtype: float64
```

In [84]:
```
# Null Hypothesis - There is no significant difference in Average
# Item_Outlet_Sales of different Item_Fat_Content. Both means can
# be treated equally
# Alt Hypothesis - There is significant difference in Average
# Item_Outlet_Sales of different Item_Fat_Content. Both means can
# not be treated equally

# Interpretation of Test output is based on p-value or probability
# value
# Interpretation based on p-value is REJECT NULL HYPOTHESIS or
# FAIL TO REJECT NULL (ACCEPT) HYPOTHESIS
# p-value is less than 0.05, REJECT NULL HYPOTHESIS
# p-value is greater than 0.05, FAIL TO REJECT NULL HYPOHESIS

# 0.05 means 5% Alpha(probable error rate) and 95% Confidence Level
```

In [85]:
```
# Split Data into Lowfat and regular dataframes
lowfat=bigmarttraindf[bigmarttraindf.Item_Fat_Content==0]
regular=bigmarttraindf[bigmarttraindf.Item_Fat_Content==1]
```

In [86]:
```
from scipy.stats import ttest_ind
```

In [87]:
```
ttest_ind(lowfat.Item_Outlet_Sales,regular.Item_Outlet_Sales,
          equal_var=False)
# Since pvalue=0.08526339464728244 is greater than 0.05, FAIL to
# REJECT NULL. Difference in means(66.8) is statistically not
# significant and it can be treated as non critical variable
```

Out[87]:
```
Ttest_indResult(statistic=nan, pvalue=nan)
```

In [88]:
```
# 2 Sample Independent ttest
# a) groupby() - mean & var
# b) Frame Null & Alt Hypothesis
# c) Split Data into Levels or Classes
# d) Conduct test
# e) Interprent result on p-value
```

In [89]:
```
# Test Null Average Item_Visibility for different Item_Fat_Content
# equal?
bigmarttraindf.Item_Visibility.groupby(
    bigmarttraindf.Item_Fat_Content).mean()
```

```
Item_Fat_Content
```

Out[89]:
```
0    0.064269
1    0.069035
Name: Item_Visibility, dtype: float64
```

In [90]:
```
bigmarttraindf.Item_Visibility.groupby(
    bigmarttraindf.Item_Fat_Content).var()
```

Out[90]:
```
Item_Fat_Content
0    0.002579
1    0.002760
Name: Item_Visibility, dtype: float64
```

In [91]:
```
ttest_ind(lowfat.Item_Visibility,regular.Item_Visibility,
          equal_var=False)
# Since pvalue=1.404696156360228e-05 is less than 0.05, REJECT NULL
# Hypothesis. Difference in means (0.0005) is statistically
# significant and variable is critical variable
```

Out[91]:
```
Ttest_indResult(statistic=-5.229251870578357, pvalue=1.7362853855379262e-07)
```

In [92]:
```
np.round(1.404696156360228e-05,)# 0.00001404696
```

Out[92]:
```
0.0
```

In [93]:
```
# Test Null Average MRP of different Item_Fat_Content are equal?
bigmarttraindf.Item_MRP.groupby(
    bigmarttraindf.Item_Fat_Content).mean()
```

Out[93]:
```
Item_Fat_Content
0    141.189424
1    140.667431
Name: Item_MRP, dtype: float64
```

In [94]:
```
bigmarttraindf.Item_MRP.groupby(
    bigmarttraindf.Item_Fat_Content).var()
```

Out[94]:
```
Item_Fat_Content
0    3855.465236
1    3854.140113
Name: Item_MRP, dtype: float64
```

In [95]:
```
ttest_ind(lowfat.Item_MRP,regular.Item_MRP,equal_var=False)
# Since pvalue=0.5755481942929463 is greater than 0.05, FAIL to
# REJECT NULL. Difference in means (0.80) is not significant and
# variable is not critical
```

Out[95]:
```
Ttest_indResult(statistic=0.4789803788348493, pvalue=0.6319627369791894)
```

In [96]:
```
# If one variable is numerical and other variable is categorical
# with more than 2 levels or classes - Anova Single Factor or
# One Way Anova

# Test Null Average Item_Outlet_Sales for different Outlet_Type is
# equal
# groupby()-mean,Null & Alt Hypo, Split Data, Conduct test, interpret
# Interpretation ia based on p-value similar to above
```

```
bigmarttraindf.Item_Outlet_Sales.groupby(
    bigmarttraindf.Outlet_Type).mean()
# 0- Grcery Store, 1 - Super Market Type1 , 2 - Super Market Type1
# 3 - Super Market Type3
```

Out[96]:
```
Outlet_Type
0     339.828500
1    2316.181148
2    1995.498739
3    3694.038558
Name: Item_Outlet_Sales, dtype: float64
```

In [97]:
```
bigmarttraindf.Item_Outlet_Sales.groupby(bigmarttraindf.Outlet_Size).mean()
```

Out[97]:
```
Outlet_Size
0    2298.995256
1    2681.603542
2    1822.626947
3    1912.149161
Name: Item_Outlet_Sales, dtype: float64
```

In [98]:
```
# Null - There is no Significant difference in Average Item outlet
# Sales for different Outlet Type
# Alt - There is Significant difference in Average Item outlet
# Sales for different Outlet Type
```

In [99]:
```
gs=bigmarttraindf[bigmarttraindf.Outlet_Type==0]
st1=bigmarttraindf[bigmarttraindf.Outlet_Type==1]
st2=bigmarttraindf[bigmarttraindf.Outlet_Type==2]
st3=bigmarttraindf[bigmarttraindf.Outlet_Type==3]
```

In [100…
```
from scipy.stats import f_oneway
```

In [101…
```
f_oneway(gs.Item_Outlet_Sales,st1.Item_Outlet_Sales,
        st2.Item_Outlet_Sales,st3.Item_Outlet_Sales)
# Since pvalue=0.0 is less than 0.05, REJECT NULL. Difference in means
# is statistically Significant
```

Out[101…
```
F_onewayResult(statistic=nan, pvalue=nan)
```

In [102…
```
# Test Null Average Item_Visibility of different Outlet_Type are
# equal?
# groupby()-mean, Null & Alt, Conduct test, interpret
bigmarttraindf.Item_Visibility.groupby(
    bigmarttraindf.Outlet_Type).mean()
```

Out[102…
```
Outlet_Type
0    0.104596
1    0.060383
2    0.059976
3    0.060344
Name: Item_Visibility, dtype: float64
```

In [103…
```
# Null -There is no significant difference in Average Item_Visibility
# of different Outlet_Type
```

```
# Alt -There is significant difference in Average Item_Visibility
# of different Outlet_Type
```

In [106…

```
f_oneway(gs.Item_Visibility,st1.Item_Visibility,
        st2.Item_Visibility,st3.Item_Visibility)
# Since pvalue=4.5932234924663505e-158 is less than 0.05, REJECT
# Null. Difference in Means is Statistically SignificantF = MST/MSE MST = SST/ p-1 M
```

Out[106…

```
F_onewayResult(statistic=423.4106805573595, pvalue=1.800566800143938e-263)
```

In [107…

```
# If one variable is numerical and Other variable is categorical
# with exactly 2 levels but levels will be Before & After an event
# Paired Sample t test or Related Sample ttest

# Null - There is no significant difference in average value before
# after
# Alt - There is significant difference in average value before
# after
```

In [108…

```
# Test Null Average CPI of India before and after demonitization
# is equal.
beforedemon=[3.35,4.14,5.30,6.46,6.13,6.59,5.86,5.51,5.53,5.91,6.32,6.72]
afterdemon=[2.59,2.23,1.86,2.62,2.62,2.21,1.09,1.08,1.79,2.52,2.89,3.24]
```

In [109…

```
print(np.mean(beforedemon))
print(np.mean(afterdemon))
```

```
5.651666666666667
2.2283333333333335
```

In [111…

```
from scipy.stats import ttest_rel
```

In [112…

```
ttest_rel(beforedemon,afterdemon)
# Since pvalue=3.607160215114515e-07 is less than 0.05, REJECT NULL
# Difference in means is statistically significant
```

Out[112…

```
Ttest_relResult(statistic=10.739765011583762, pvalue=3.607160215114515e-07)
```

In [113…

```
ttest_rel(beforedemon,afterdemon)
# Since pvalue=3.607160215114515e-07 is less than 0.05, REJECT NULL
# Difference in means is statistically significant
```

Out[113…

```
Ttest_relResult(statistic=10.739765011583762, pvalue=3.607160215114515e-07)
```

In [114…

```
# Both Variables are non numeric and categorical - Chi Square Test
# of Independence
# Input of Chi Square Test of Independence is Cross Tabulation
# Cross tabulation is frequency table of 2 non numeric categorical
# variables
pd.crosstab(bigmarttraindf.Outlet_Type,bigmarttraindf.Outlet_Size)
# Outlet_Type -  0- Grcery Store, 1 - Super Market Type1 ,
# 2 - Super Market Type1 3 - Super Market Type3
# Outlet_Size - 0- High 1 -Medium 2- Missing 3 - Small
```

Out[114...

Outlet_Size	0	1	2	3
Outlet_Type				
0	0	0	925	880
1	1553	1550	3091	3100
2	0	1546	0	0
3	0	1559	0	0

In [115...

```
# Null - There is no Association/Relationship between both variables
# Alt - There is Association/Relationship between both variables
```

In [116...

```
from scipy.stats import chi2_contingency
```

In [117...

```
chi2_contingency(pd.crosstab(bigmarttraindf.Outlet_Type,
                             bigmarttraindf.Outlet_Size))
# Since p-value=0.0, is less than 0.05, REJECT NULL
```

Out[117...

```
(8966.446215630342,
 0.0,
 9,
 array([[ 197.35039426,  591.54287525,  510.34074908,  505.76598141],
        [1016.16319347, 3045.8722895 , 2627.76006759, 2604.20444945],
        [ 169.03252605,  506.66220783,  437.11179949,  433.19346663],
        [ 170.45388623,  510.92262743,  440.78738384,  436.83610251]]))
```

In [118...

```
# Test Null There is no Association or Relationship between
# Outlet_Type and Outlet_Location_Type
```

In [119...

```
pd.crosstab(bigmarttraindf.Outlet_Location_Type,
            bigmarttraindf.Outlet_Type)
```

Out[119...

Outlet_Type	0	1	2	3
Outlet_Location_Type				
0	880	3100	0	0
1	0	4641	0	0
2	925	1553	1546	1559

In [120...

```
chi2_contingency(pd.crosstab(bigmarttraindf.Outlet_Location_Type,
            bigmarttraindf.Outlet_Type))
# Since p-value=0 REJECT NULL
```

Out[120...

```
(7875.9685214158635,
 0.0,
 6,
 array([[ 505.76598141, 2604.20444945,  433.19346663,  436.83610251],
        [ 589.76379893, 3036.71177133,  505.13841172,  509.38601802],
        [ 709.47021966, 3653.08377922,  607.66812166,  612.77787947]]))
```

In [121...

```
# Machine Learning is training machines or computers on historical data
```

```
# and use them for making predictions on current data or future data.

# Machine Learning uses statistical methodologies but do not strictly
# adhere to statistical assumptions.

# Machine Learning - Supervised learning and Unsupervised Learning
# Supervised Learning - Understand data thoroughly and also know which
# machine learning models to implement.
# Unsupervised Learning - No understanding of data, and do not know which
# machine learning models to implement.

# Supervised Learning -
# 1) Regression Techniques - Dependent Variable(y) is nuemrical continuous.
# closer to normal distribution with minimum outliers and no missing
# values.
# 2) Classification Techniques - Dependent Variable(y) is non numeric
# either Binary (yes/no) or multinomial(more than 2 levels or groups)
```

In [122…

```
# Supervised Learning - Regression techniques - Multiple Linear Regression
# Multiple Linear Regression explains the linear relationship between a
# dependent variable(y) and multiple independent variables(X). Linear
# relationship indicates straight line relationship.

# Multiple Linear Regression is an equation
# y = B0 + B1X1 + B2X2 + B3X3 + B4X4 + ..... + BnXn + Et
# y - Dependent Variable
# B0 - intercept or constant. Point at which straight line touches y-axis
# B1, B2, B3, B4,.... Bn - Coefficients
# X1, X2, X3, X4,.....Xn - Independent Variables
# Et - Residual = ActualValue - Predicted Value or Error term

# In a MLR, we have data for y and data of all X's. We do not have the
# intercept and coefficients. Algorithm will solve for the intercept
# and coefficients.
```

In [123…

```
# MLR is an parametric algorithm that has many assumptions or high bias
# to be met before implementing algorithm.
# Assumptions
# 1) Dependent Variable(y) must be numerical, continuos, closer to normal
# distrinution, with minimum outliers and no missing values.
# 2) Independent Variables can be both numerical and non nuemrical categorical
# 3) Linearity - There must be a logical linear relationship between
# dependent variable and independent variables.
# 4) No Multicollinearity - Multicollinearity means very strong correlation
# greater than 0.95 between variables. If multicollinearity exists do
# not include both variables choose one among them. Varaiance Inflation
# Factor (VIF) greater than 2 indicates multicollinearity among variables
# 5) Exogenity - Dependent variable is dependent on Independent variables
# but not vice-versa. y is dependent on X's but X's do not depend on y.
# 6) Sample size required is minimum 20 observations per variable

# Post Model Assumptions on Residuals
# 7) Residuals must be normally distributed. Check using normal q-q plot
# 8) Residuals must be homoscedastic pattern.

# Interpretation of Output in Python only(Non Statistical Output)
# 1) R Square must be between 0.60 - 0.95. R Square explains the amount of
# variance occuring in the dependent variable casued by the model or
# independent Variables.

# 2 ) Intercept + Coefficients - Using which a regression equation will be
```

```
# built and it will be used for predicting values (yhat)

# Diagnostic for all Regression Models
# Root Mean Square Error (RMSE) - There is no fixed range but the model
# that has least RMSE is best fit model.
```

In [124…

```
bigmarttraindf.columns
```

Out[124…

```
Index(['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Item_Outlet_Sales',
       'Outlet_Age', 'Item_Fat_Content', 'Item_Type', 'Outlet_Identifier',
       'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type',
       'Item_outlet_Sales', 'Item_Code'],
      dtype='object')
```

In [125…

```
# Correlation Analysis must be done on Numerical Data
bigmarttraindf[['Item_Weight', 'Item_Visibility', 'Item_MRP',
                'Outlet_Age','Item_Outlet_Sales']].corr()
```

Out[125…

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Age	Item_Outlet_Sales
Item_Weight	1.000000	-0.013641	0.033002	-0.000462	0.011083
Item_Visibility	-0.013641	1.000000	-0.006351	0.083678	-0.128625
Item_MRP	0.033002	-0.006351	1.000000	-0.000141	0.567574
Outlet_Age	-0.000462	0.083678	-0.000141	1.000000	0.049135
Item_Outlet_Sales	0.011083	-0.128625	0.567574	0.049135	1.000000

In [126…

```
# Split data into Dependent Variable(y) and Independent variables(X)
y=bigmarttraindf.Item_Outlet_Sales
X=bigmarttraindf.drop('Item_Outlet_Sales',axis=1)
```

In [127…

```
from sklearn.linear_model import LinearRegression
```

In [142…

```
regmodel=LinearRegression()
```

In [147…

```
# Regression Equation for prediction is
# Item_Outlet_Sales = -891.28 - 0.56*Item_Weight -1627.96 * Item_Visibility
# + 15.57 * Item_MRP +1.315*Outlet_Age + 54.34* Item_fat_Content
# -0.93*Item_Type + 56.47*Outlet_Identifier -93.01*Outlet_Size
# -122.16 * outlet_location_type + 831 * Outlet_type -18.31* Item_code
```

In [ ]: