Electronics Sales Prediction

Installs

In [4]:
```
!pip install -q autoviz
!pip install -q -U --pre pycaret
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: htt
ps://pip.pypa.io/warnings/venv
  WARNING: Requested plotly-resampler>=0.7.2.2 from https://files.pythonhosted.org/packages/d7/5e/71a9e34a36c1855d0c4e30a88405d58c4bbbe7ece802b188628a643f2cda/plotly_resampler-0.8.4rc1.tar.gz
(from pycaret), but installing version 0.8.4rc1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
pydocstyle 6.2.3 requires importlib-metadata<5.0.0,>=2.0.0; python_version < "3.8", but you have importlib-metadata 6.0.0 which is incompatible.
librosa 0.10.0 requires soundfile>=0.12.1, but you have soundfile 0.11.0 which is incompatible.
ibis-framework 2.1.1 requires importlib-metadata<5,>=4; python_version < "3.8", but you have importlib-metadata 6.0.0 which is incompatible.
flake8 5.0.4 requires importlib-metadata<4.3,>=1.1.0; python_version < "3.8", but you have importlib-metadata 6.0.0 which is incompatible.
cmudict 1.0.13 requires importlib-metadata<6.0.0,>=5.1.0, but you have importlib-metadata 6.0.0 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: htt
ps://pip.pypa.io/warnings/venv
```

Imports

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats
from pandas_profiling import ProfileReport
from statsmodels.stats.outliers_influence import variance_inflation_factor
from autoviz.classify_method import data_cleaning_suggestions ,data_suggestions

from pycaret  import regression
from sklearn.model_selection import cross_val_score
```

Data Loading

In [7]:
```
df = pd.read_csv('/content/sample_data/Advertising.csv')
```

In [8]:
```
df.head()
```

Out[8]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [9]:
```
df.tail()
```

Out[9]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

In [13]:
```
df.drop('Unnamed: 0', axis = 1, inplace = True)
```

EDA

In [14]:
```
df.shape
```

Out[14]:
```
(200, 4)
```

In [15]:
```
data_cleaning_suggestions(df)
```

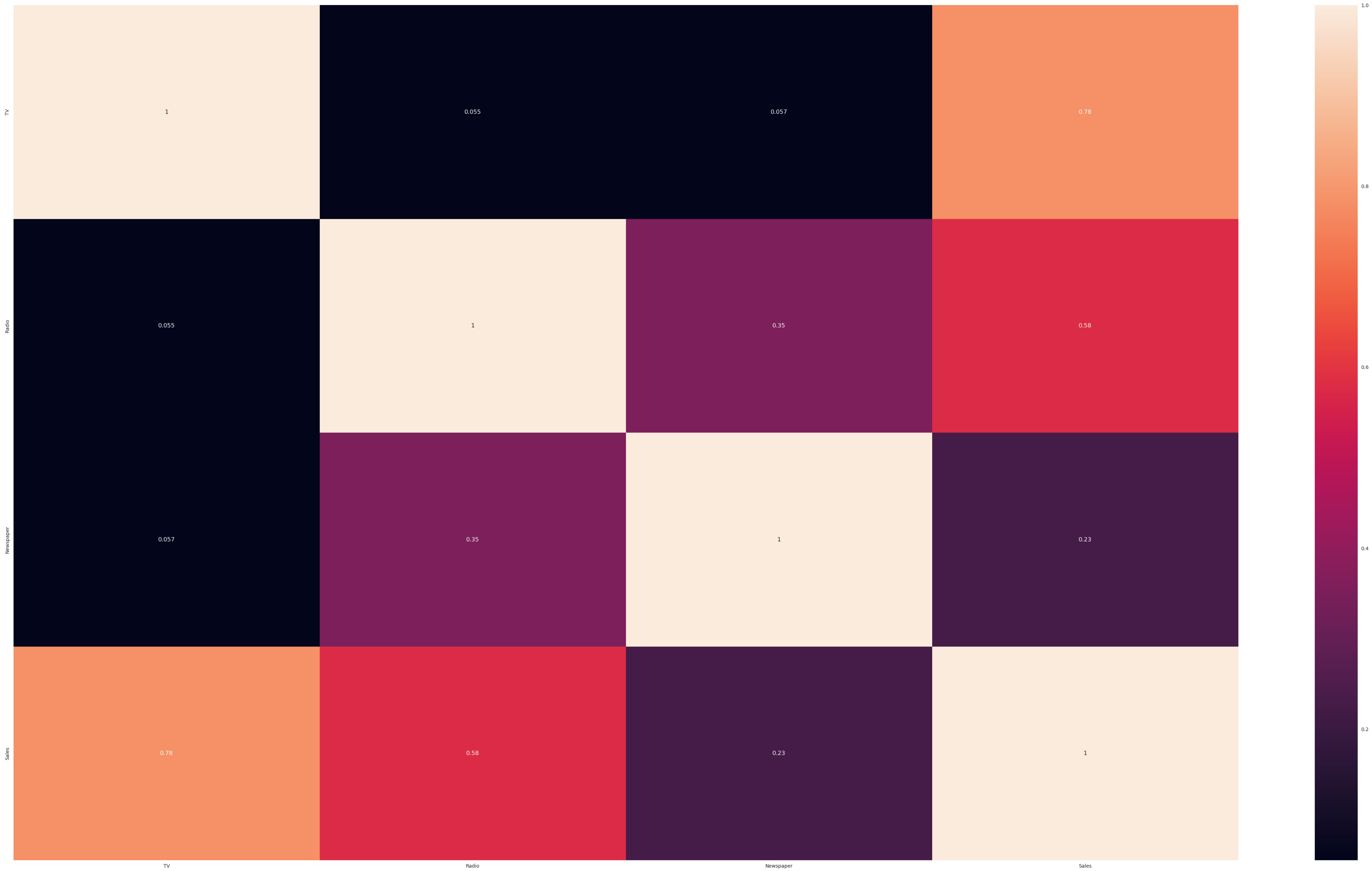
```
Data cleaning improvement suggestions. Complete them before proceeding to ML modeling.
```

	Nuniques	dtype	Nulls	Nullpercent	NuniquePercent	Value counts Min	Data cleaning improvement suggestions
TV	190	float64	0	0.000000	95.000000	0	
Newspaper	172	float64	0	0.000000	86.000000	0	
Radio	167	float64	0	0.000000	83.500000	0	
Sales	121	float64	0	0.000000	60.500000	0	

Correlation

In [16]:
```
plt.figure(figsize=(50,30))
sns.heatmap(df.corr(),annot=True)
```

Out[16]:
```
<AxesSubplot:>
```


TV
Radio
Newspaper
Sales
1
0.055
0.057
0.78
0.055
1
0.35
0.58
0.057
0.35
1
0.23
0.78
0.58
0.23
1
1.0
0.8
0.6
0.4
0.2

Outliers

In [17]:
```
def detect_outliers(data):
    outlier_percents = {}
    for column in data.columns:
        if data[column].dtype != object:
            q1 = np.quantile(data[column], 0.25)
            q3 = np.quantile(data[column], 0.75)
            iqr = q3 - q1
            upper_bound = q3 + (1.5 * iqr)
            lower_bound = q1 - (1.5 * iqr)
            outliers = data[(data[column] > upper_bound) | (data[column] < lower_bound)][column]
            outlier_percentage = len(outliers) / len(data[column]) * 100
            outlier_percents[column] = outlier_percentage
            outlier_dataframe = pd.DataFrame(data = outlier_percents.values() ,index=outlier_percents.keys() ,columns=['Outlier_percentage'])

    return outlier_dataframe.sort_values(by = 'Outlier_percentage', ascending = False)

detect_outliers(df)
```

Out[17]:

	Outlier_percentage
Newspaper	1.0
TV	0.0
Radio	0.0
Sales	0.0

Comparing Regression Models

In [18]:
```
from pycaret.regression import *
```

In [20]:
```
X = df.drop('Sales', axis = 1)
y = df['Sales']
```

In [21]:
```
s = setup(data = df, target = 'Sales', session_id=123)
```

	Description	Value
0	Session id	123
1	Target	Sales
2	Target type	Regression
3	Original data shape	(200, 4)
4	Transformed data shape	(200, 4)
5	Transformed train set shape	(140, 4)
6	Transformed test set shape	(60, 4)
7	Numeric features	3
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	KFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	reg-default-name
18	USI	99d8

In [22]:
```
compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.4624	0.4099	0.6207	0.9829	0.0638	0.0524	0.1550
rf	Random Forest Regressor	0.6819	0.7522	0.8463	0.9707	0.0810	0.0698	0.1820
gbr	Gradient Boosting Regressor	0.6543	0.7543	0.8449	0.9695	0.0821	0.0685	0.0550
catboost	CatBoost Regressor	0.5632	0.9416	0.8862	0.9636	0.1010	0.0774	0.6820
xgboost	Extreme Gradient Boosting	0.7092	0.8571	0.9083	0.9628	0.0842	0.0708	0.1400
ada	AdaBoost Regressor	0.9118	1.3620	1.1184	0.9487	0.1028	0.0923	0.0550
dt	Decision Tree Regressor	0.9336	1.6092	1.2321	0.9283	0.1167	0.0949	0.0250
lightgbm	Light Gradient Boosting Machine	0.9731	1.8903	1.3323	0.9196	0.1434	0.1178	0.1270
knn	K Neighbors Regressor	1.2407	2.7481	1.6193	0.8813	0.1221	0.1120	0.0490
lasso	Lasso Regression	1.3834	3.3047	1.7500	0.8674	0.1721	0.1612	0.0240
en	Elastic Net	1.3842	3.3202	1.7524	0.8669	0.1734	0.1621	0.0240
lar	Least Angle Regression	1.3846	3.3397	1.7555	0.8663	0.1752	0.1634	0.0230
lr	Linear Regression	1.3846	3.3397	1.7555	0.8663	0.1752	0.1634	0.3240
ridge	Ridge Regression	1.3846	3.3397	1.7555	0.8663	0.1752	0.1634	0.0240
br	Bayesian Ridge	1.3888	3.3451	1.7580	0.8659	0.1743	0.1631	0.0240
huber	Huber Regressor	1.3405	3.4828	1.7876	0.8618	0.1780	0.1661	0.0260
omp	Orthogonal Matching Pursuit	2.6871	11.2996	3.3293	0.4990	0.2264	0.2269	0.0250
llar	Lasso Least Angle Regression	4.3561	27.7343	5.1586	-0.0588	0.3803	0.4259	0.0240
dummy	Dummy Regressor	4.3561	27.7343	5.1586	-0.0588	0.3803	0.4259	0.0410
par	Passive Aggressive Regressor	5.0575	56.0901	6.1967	-1.0559	0.3472	0.3877	0.0250

```
Processing:   0%|          | 0/85 [00:00<?, ?it/s]
```

Out[22]:
```
ExtraTreesRegressor(n_jobs=-1, random_state=123)
```

Extra Trees Regressor Model

Extra Trees Regressor is an ensemble machine learning algorithm that is used for regression tasks. It is based on the Random Forest algorithm and works by creating a large number of decision trees, each using a random subset of the available features and data. It then combines the predictions of all these decision trees to make a final prediction. One key difference between Extra Trees and Random Forest is that Extra Trees selects the splitting thresholds of each node randomly, rather than searching for the best threshold. This makes Extra Trees faster to train than Random Forests, while still achieving good predictive performance.

In [23]:
```
et = create_model('et')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.3637	0.3478	0.5897	0.9842	0.0373	0.0286
1	0.6276	1.1538	1.0741	0.9629	0.2399	0.2001
2	0.4551	0.4245	0.6515	0.9864	0.0389	0.0328
3	0.4281	0.3163	0.5624	0.9733	0.0404	0.0326
4	0.4111	0.2964	0.5444	0.9760	0.0422	0.0323
5	0.4338	0.2968	0.5448	0.9871	0.0384	0.0359
6	0.4693	0.3032	0.5506	0.9909	0.0468	0.0407
7	0.4261	0.2542	0.5042	0.9869	0.0565	0.0431
8	0.4567	0.3008	0.5485	0.9910	0.0603	0.0414
9	0.5528	0.4052	0.6366	0.9902	0.0377	0.0365
Mean	0.4624	0.4099	0.6207	0.9829	0.0638	0.0524
Std	0.0716	0.2529	0.1570	0.0088	0.0592	0.0494

```
Processing:   0%|          | 0/4 [00:00<?, ?it/s]
```

In [24]:
```
et = finalize_model(et)
et
```

Out[24]:
```
Pipeline(memory=FastMemory(location=/tmp/joblib),
         steps=[('numerical_imputer',
                 TransformerWrapper(include=['TV', 'Radio', 'Newspaper'],
                                    transformer=SimpleImputer())),
                ('categorical_imputer',
                 TransformerWrapper(include=[],
                                    transformer=SimpleImputer(strategy='most_frequent'))),
                ('actual_estimator',
                 ExtraTreesRegressor(n_jobs=-1, random_state=123))])
```

In [25]:
```
preds = predict_model(et)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Extra Trees Regressor	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000

R2 Score : 1.0000

Predictions

In [26]:
```
preds
```

Out[26]:

	TV	Radio	Newspaper	Sales	prediction_label
140	199.800003	3.100000	34.599998	11.400000	11.400000
141	80.199997	0.000000	9.200000	8.800000	8.800000
142	74.699997	49.400002	45.700001	14.700000	14.700000
143	44.700001	25.799999	20.600000	10.100000	10.100000
144	147.300003	23.900000	19.100000	14.600000	14.600000
145	238.199997	34.299999	5.300000	20.700001	20.700001
146	165.600006	10.000000	17.600000	12.600000	12.600000
147	182.600006	46.200001	58.700001	21.200001	21.200001
148	188.399994	18.100000	25.600000	14.900000	14.900000
149	11.700000	36.900002	45.200001	7.300000	7.300000
150	75.300003	20.299999	32.500000	11.300000	11.300000
151	205.000000	45.099998	19.600000	22.600000	22.600000
152	56.200001	5.700000	29.700001	8.700000	8.700000
153	18.700001	12.100000	23.400000	6.700000	6.700000
154	13.100000	0.400000	25.600000	5.300000	5.300000
155	112.900002	17.400000	38.599998	11.900000	11.900000
156	180.800003	10.800000	58.400002	12.900000	12.900000
157	276.700012	2.300000	23.700001	11.800000	11.800000
158	18.799999	21.700001	50.400002	7.000000	7.000000
159	218.399994	27.700001	53.400002	18.000000	18.000000
160	19.600000	20.100000	17.000000	7.600000	7.600000
161	88.300003	25.500000	73.400002	12.900000	12.900000
162	17.900000	37.599998	21.600000	8.000000	8.000000
163	50.000000	11.600000	18.400000	8.400000	8.400000
164	220.300003	49.000000	3.200000	24.700001	24.700001
165	26.799999	33.000000	19.299999	8.800000	8.800000
166	156.600006	2.600000	8.300000	10.500000	10.500000
167	142.899994	29.299999	12.600000	15.000000	15.000000
168	96.199997	14.800000	38.900002	11.400000	11.400000
169	216.399994	41.700001	39.599998	22.600000	22.600000
170	116.000000	7.700000	23.100000	11.000000	11.000000
171	250.899994	36.500000	72.300003	22.200001	22.200001
172	287.600006	43.000000	71.800003	26.200001	26.200001
173	19.400000	16.000000	22.299999	6.600000	6.600000
174	193.199997	18.400000	65.699997	15.200000	15.200000

	TV	Radio	Newspaper	Sales	prediction_label
175	219.800003	33.500000	45.099998	19.600000	19.600000
176	253.800003	21.299999	30.000000	17.600000	17.600000
177	184.899994	43.900002	1.700000	20.700001	20.700001
178	163.300003	31.600000	52.900002	16.900000	16.900000
179	73.400002	17.000000	12.900000	10.900000	10.900000
180	62.299999	12.600000	18.299999	9.700000	9.700000
181	280.700012	13.900000	37.000000	16.100000	16.100000
182	78.199997	46.799999	34.500000	14.600000	14.600000
183	265.600006	20.000000	0.300000	17.400000	17.400000
184	228.300003	16.900000	26.200001	15.500000	15.500000
185	164.500000	20.900000	47.400002	14.500000	14.500000
186	177.000000	33.400002	38.700001	17.100000	17.100000
187	222.399994	4.300000	49.799999	11.700000	11.700000
188	197.600006	23.299999	14.200000	16.600000	16.600000
189	109.800003	14.300000	31.700001	12.400000	12.400000
190	139.500000	2.100000	26.600000	10.300000	10.300000
191	225.800003	8.200000	56.500000	13.400000	13.400000
192	293.600006	27.700001	1.800000	20.700001	20.700001
193	141.300003	26.799999	46.200001	15.500000	15.500000
194	75.500000	10.800000	6.000000	9.900000	9.900000
195	85.699997	35.799999	49.299999	13.300000	13.300000
196	66.099998	5.800000	24.200001	8.600000	8.600000
197	276.899994	48.900002	41.799999	27.000000	27.000000
198	120.500000	28.500000	14.200000	14.200000	14.200000
199	239.300003	15.500000	27.299999	15.700000	15.700000