



DATA ANALYTICS

UE19CS312

PHASE 1 REPORT

Analysis of Camera Dataset using Machine Learning Algorithms

Team Data Pirates

Team Member Name	SRN
Phani Kumar Vedurumudi	PES2UG19CS281
Pawan Prasad P	PES2UG19CS280
G U Deepak	PES2UG19CS124
Jayanth O	PES2UG19CS163

Dataset Name and Description:

Dataset Name:

1000 Cameras Dataset

Link: <https://www.kaggle.com/crawford/1000-cameras-dataset>

Dataset Description:

Data has been collected from Kaggle's dataset which is online and free dataset. 1000 Cameras Dataset have been gathered and cleaned up by Petra Isenberg, Pierre Dragicevic and Yvonne Jansen.

DATASET SELECTION:

1. Dataset name: Camera Dataset
2. Source: Kaggle
3. No. of Observations: 1038
4. No. of Columns: 13
5. Percentage of Missing Values: 5%

This Dataset contains all the details of different Camera Models features and their values which helps to analysis various real-world problems.

This Dataset contains a combination of numerical and categorical features:

Categorical:

- Model

Numerical:

- Release date
- Max resolution
- Low resolution
- Effective pixels
- Zoom wide (W)
- Zoom tele (T)
- Normal focus range
- Macro focus range
- Storage included
- Weight (inc. batteries)
- Dimensions
- Price

Describe the features of the dataset:

1. **MODEL:** This feature of the dataset describes about the various models of the different type(company) of cameras present in the chosen dataset.
2. **RELEASE DATE:** This feature of the dataset describes about the various release dates (in years) of the different models of the cameras in the dataset.
3. **MAX RESOLUTION:** This feature of the dataset describes about the maximum resolution of the different models of the cameras present in the dataset.
4. **MIN RESOLUTION:** This feature of the dataset describes about the minimum resolution of the different models of the cameras present in the dataset.
5. **EFFECTIVE PIXELS:** This feature of the dataset describes about the different number of effective pixels of each camera model in the dataset.
6. **ZOOM WIDE:** This feature of the dataset describes about the short focal length of the different cameras in the dataset.
7. **ZOOM TELE:** This feature of the dataset describes about the long focal length of the different cameras in the dataset.
8. **NORMAL FOCUS RANGE:** This feature of the dataset describes about the lens of the various cameras in the dataset with a focal length approximately equal to the diagonal of the film format or of a digital camera's image sensor. Most 35mm cameras normal lenses have a focal length of approximately 50 mm.
9. **MACRO FOCUS RANGE:** This feature of the dataset describes about the focal length of the various cameras in the dataset, a macro lens (50 mm on a 35 mm camera) can focus so close that lighting remains difficult. Macro lens has a focal length from about 100 to 200 mm.
10. **STORAGE INCLUDED:** This feature of the dataset describes about the inbuilt storage that is included in the various cameras of the dataset.
11. **WEIGHT:** This feature of the dataset describes about the weight of the cameras (inclusive of the weight of the batteries) in the dataset.
12. **DIMENSION:** This feature of the dataset describes about the dimensions of the cameras in the dataset.
13. **PRICE:** This feature of the dataset describes about the prices of the cameras in the dataset.

Problem statement:

Our project objective is to predict the camera model based on the given features and price of the camera. We will use the machine learning algorithms of KNN(k-nearest neighbours), Logistic Regression and SVM.

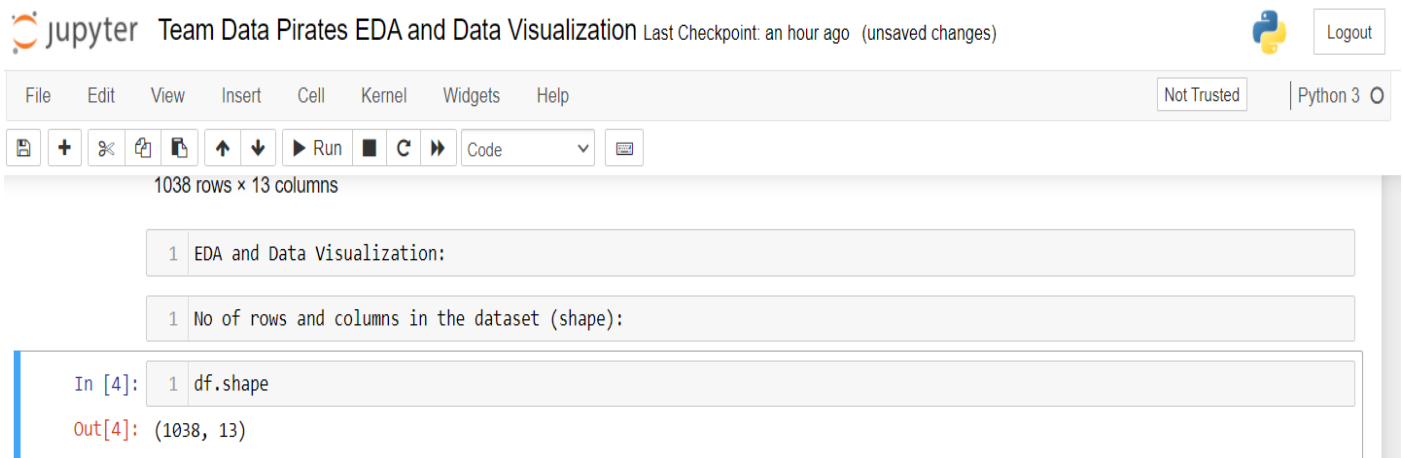
EDA and Data Visualization:

We have use **Jupyter Notebook** for this task.

🚀 How many rows and attributes?

No. of Rows: 1038

No. of Attributes: 13



The screenshot shows a Jupyter Notebook interface with the following elements:

- Header:** "jupyter Team Data Pirates EDA and Data Visualization" with a "Last Checkpoint: an hour ago (unsaved changes)" status and a "Logout" button.
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Toolbar:** Includes icons for saving, opening, and running code, along with a "Code" dropdown menu.
- Code Cells:**
 - Cell 1: "EDA and Data Visualization:"
 - Cell 2: "No of rows and columns in the dataset (shape):"
- Output:** Below the second cell, the output shows:

```
In [4]: 1 df.shape
Out[4]: (1038, 13)
```

🚀 How many missing data and outliers?

Number of missing values in the dataset attribute-wise: 689

Total percentage of missing values in the dataset: 5.10 %

Out[4]: (1000, 15)

1 Number of missing values in the dataset attribute-wise:

In [7]: 1 df.isnull().sum()

Out[7]:

Model	0
Release date	0
Max resolution	1
Low resolution	54
Effective pixels	35
Zoom wide (W)	85
Zoom tele (T)	85
Normal focus range	137
Macro focus range	128
Storage included	125
Weight (inc. batteries)	23
Dimensions	16
Price	0
dtype: int64	

Total number of missing values in the dataset:

In [6]: 1 df.isnull().sum().sum()

Out[6]: 689

In []: 1 Total percentage of missing values in the dataset:

In [8]: 1 (df.isnull().sum().sum()/(df.shape[0]*df.shape[1]))*100

Out[8]: 5.105973025048169



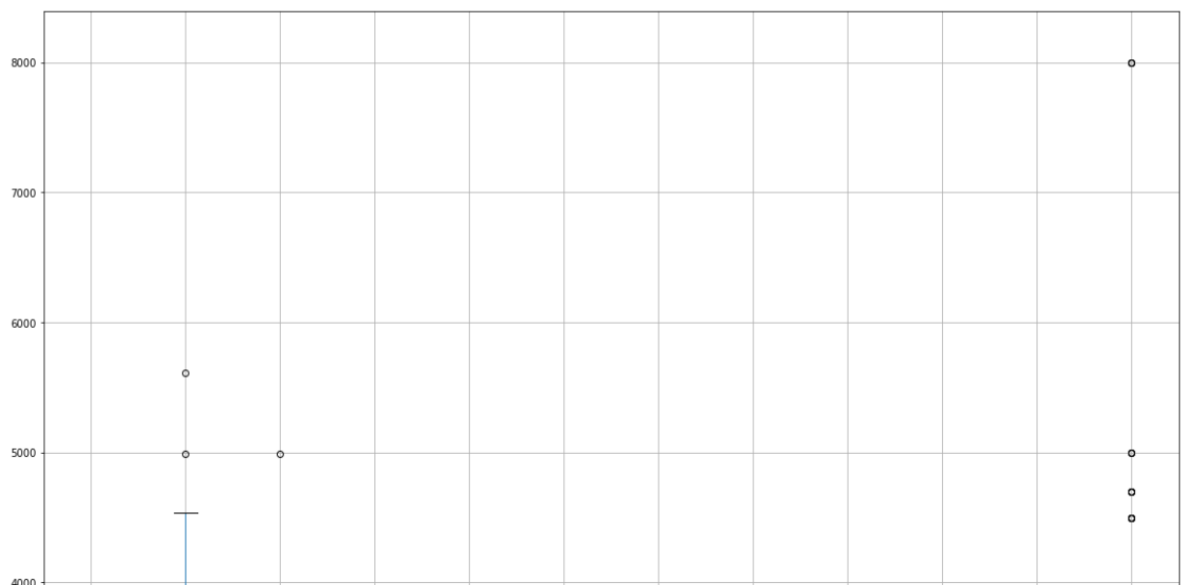
Logout

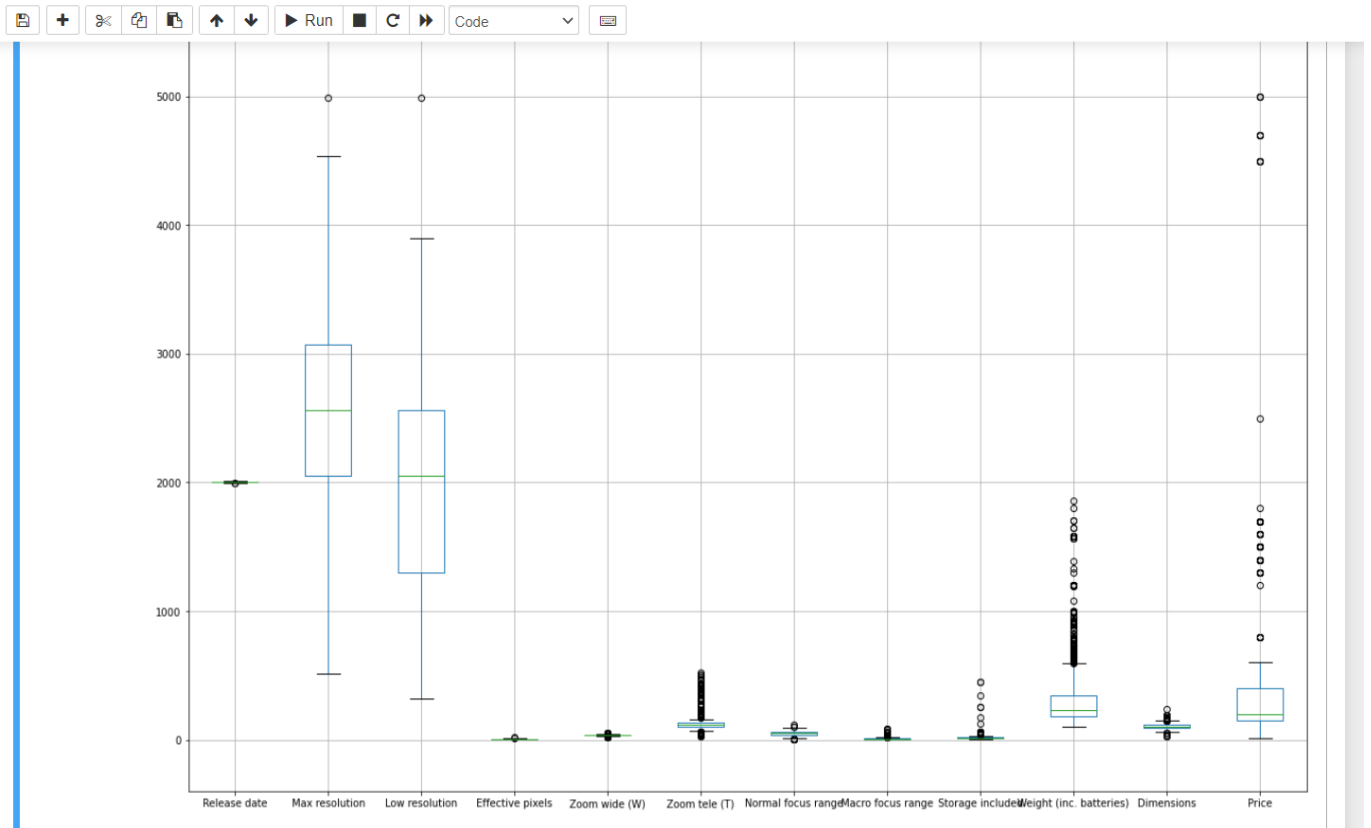
Outliers:

Visualisation of Outliers in the dataset:

In [13]: 1 df.boxplot(figsize=(20,20))

Out[13]: <AxesSubplot:>





Finding the No. of outliers in each of the columns/features in the Dataset:

```
In [14]: 1 outliers=dict()
2 for i in df.columns:
3     if i!='Model':
4         Q1=df[i].quantile(q=0.25,interpolation='midpoint')
5         Q3=df[i].quantile(q=0.75,interpolation='midpoint')
6         IQR=Q3-Q1
7         for j in df[i]:
8             if j>Q3+1.5*IQR or j<Q1-1.5*IQR:
9                 if i not in outliers:
10                    outliers[i]=0
11                else:
12                    outliers[i]+=1
13 print(outliers)
```

```
{'Release date': 1, 'Max resolution': 1, 'Low resolution': 0, 'Effective pixels': 1, 'Zoom wide (W)': 89, 'Zoom tele (T)': 281,
'Normal focus range': 14, 'Macro focus range': 102, 'Storage included': 37, 'Weight (inc. batteries)': 115, 'Dimensions': 49,
'Price': 143}
```

There is no need to remove the outliers for our dataset because it is valid numerical values which describe dataset correctly.

There is no inconsistent data and no incorrect data in the dataset.

Any inconsistent, incomplete, duplicate or incorrect data?

```
In [ ]: 1 Handling of missing values in the dataset(Incomplete Data): |
        2 We have used the mean(average) of each column to replace the NaN values of that column.
```

```
In [9]: 1 for i in df:
        2     if(i=="Model"):
        3         continue;
        4     if(i=="Price" or i=="Weight (inc. batteries)"):
        5         df[i].fillna(df[i].mean(),inplace=True)
        6     else:
        7         df[i].fillna(int(df[i].mean()),inplace=True)
        8
```

```
In [10]: 1 df
```

Out[10]:

	Model	Release date	Max resolution	Low resolution	Effective pixels	Zoom wide (W)	Zoom tele (T)	Normal focus range	Macro focus range	Storage included	Weight (inc. batteries)	Dimensions	Price
0	Agfa ePhoto 1280	1997	1024.0	640.0	4.0	38.0	114.0	70.0	40.0	4.0	420.000000	95.0	179
1	Agfa ePhoto 1680	1998	1280.0	640.0	1.0	38.0	114.0	50.0	8.0	4.0	420.000000	158.0	179
2	Agfa ePhoto CL18	2000	640.0	1871.0	4.0	45.0	45.0	50.0	8.0	2.0	325.870936	106.0	179
3	Agfa ePhoto CL30	1999	1152.0	640.0	4.0	35.0	35.0	50.0	8.0	4.0	325.870936	106.0	269
4	Agfa ePhoto CL30 Click!	1999	1152.0	640.0	4.0	43.0	43.0	50.0	8.0	40.0	300.000000	128.0	1299
...

Team Data Pirates EDA and Data Visualization | 1000 Cameras Dataset | Kaggle | Team Data Pirates Literature Survey | +

st:8889/notebooks/Downloads/Team%20Data%20Pirates%20EDA%20and%20Data%20Visualization.ipynb

jupyter Team Data Pirates EDA and Data Visualization Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [11]: 1 df.isnull().sum()
```

```
Out[11]: Model                0
         Release date         0
         Max resolution        0
         Low resolution        0
         Effective pixels      0
         Zoom wide (W)         0
         Zoom tele (T)         0
         Normal focus range    0
         Macro focus range     0
         Storage included      0
         Weight (inc. batteries) 0
         Dimensions           0
         Price                 0
         dtype: int64
```

There are no duplicate values/rows in the dataset

```
In [12]: 1 df.duplicated().sum()
```

```
Out[12]: 0
```


Are any of the preprocessing techniques needed: dimensionality reduction, range transformation, standardization, etc.?

Scatter plot:

jupyter Team Data Pirates EDA and Data Visualization Last Checkpoint: 5 hours ago (autosaved)

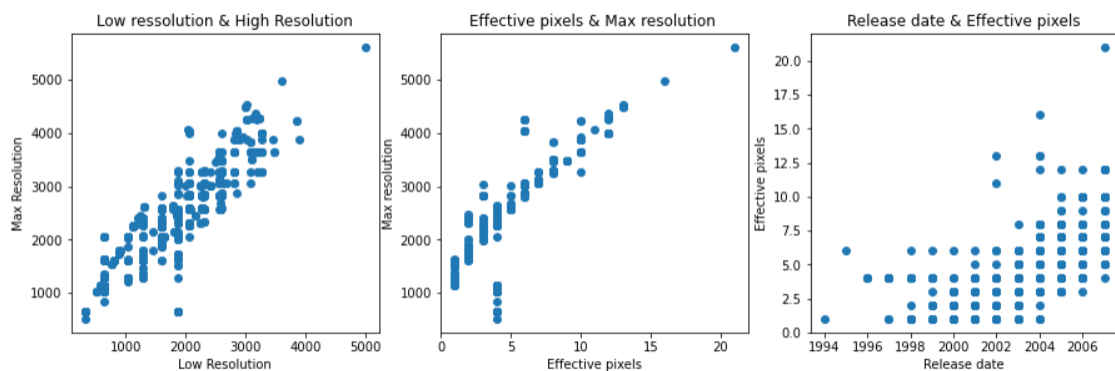


Edit View Insert Cell Kernel Widgets Help

Not Trusted

+ < > Run Code

```
In [17]: 1 fig= plt.figure(figsize=(15,15))
2 ax2=fig.add_subplot(331)
3 plt.scatter(df['Low resolution'], df['Max resolution'])
4 plt.title('Low resolution & High Resolution')
5 plt.xlabel('Low Resolution')
6 plt.ylabel('Max Resolution')
7 ax2=fig.add_subplot(332)
8 plt.scatter(df['Effective pixels'], df['Max resolution'])
9 plt.title('Effective pixels & Max resolution')
10 plt.xlabel('Effective pixels')
11 plt.ylabel('Max resolution')
12 ax2=fig.add_subplot(333)
13 plt.scatter(df['Release date'], df['Effective pixels'])
14 plt.title('Release date & Effective pixels')
15 plt.xlabel('Release date')
16 plt.ylabel('Effective pixels')
17 plt.show()
```



jupyter Team Data Pirates EDA and Data Visualization Last Checkpoint: 5 hours ago (autosaved)



Logout

Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

+ < > Run Code

Data Preprocessing: Normalization

```
In [46]: 1 numeric_dataset = df[['Release date', 'Max resolution', 'Low resolution', 'Effective pixels', 'Zoom wide (W)', 'Zoom tele (T)', 'N
2 df=(numeric_dataset-numeric_dataset.mean())/numeric_dataset.std()
```

```
In [48]: 1 df
```

```
Out[48]:
```

	Release date	Max resolution	Low resolution	Effective pixels	Zoom wide (W)	Zoom tele (T)	Normal focus range	Macro focus range	Storage included	Weight (inc. batteries)	Dimensions	Price
0	-2.418771	-1.923022	-1.711533	-0.269358	0.692387	-0.212910	1.137977	4.130904	-0.589035	3.675801e-01	-0.565086	-0.366077
1	-2.051766	-1.584224	-1.711533	-1.374472	0.692387	-0.212910	-0.044011	-0.101425	-0.589035	3.675801e-01	2.453354	-0.366077
2	-1.317755	-2.431219	-0.000378	-0.269358	2.925357	-1.014183	-0.044011	-0.101425	-0.664061	-1.109886e-15	-0.038057	-0.366077
3	-1.684760	-1.753623	-1.711533	-0.269358	-0.264600	-1.130310	-0.044011	-0.101425	-0.589035	-1.109886e-15	-0.038057	-0.247727
4	-1.684760	-1.753623	-1.711533	-0.269358	2.287365	-1.037409	-0.044011	-0.101425	0.761420	-1.010277e-01	1.016001	1.106729
...
1033	-0.950749	-0.567829	-1.177753	-0.637729	0.692387	-0.212910	-2.407989	0.163096	-0.438985	-2.292639e-02	0.632707	-0.519933
1034	-1.317755	-0.567829	-1.177753	-0.637729	-0.264600	-0.317424	1.728971	0.030835	-0.138884	2.504282e-01	0.441060	-0.519933
1035	-0.950749	-0.567829	-1.177753	-0.637729	-0.264600	-0.398712	1.728971	0.163096	-0.438985	5.517491e-02	0.009855	-0.519933
1036	-0.950749	-0.101982	-0.933103	-0.637729	-0.264600	-0.398712	1.728971	0.163096	-0.138884	5.517491e-02	0.009855	-0.519933
1037	-0.583744	-1.160726	-1.489125	-1.374472	0.692387	-1.095472	-0.635006	1.485699	-0.438985	-5.696355e-01	-0.996292	-0.431827

1038 rows x 12 columns

Team Members:
Pawan Prasad P
Phani Kumar Vedurumudi
Jayanth O
G U Deepak



Logout

Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

+ %< > Run Code

Covariance:
The measure of how the variables vary with respect to one another

In [18]: 1 df.cov()

Out[18]:

	Release date	Max resolution	Low resolution	Effective pixels	Zoom wide (W)	Zoom tele (T)	Normal focus range	Macro focus range	Storage included	Weight (inc. batteries)	Dimensions
Release date	7.424289	1633.146553	1545.303242	5.380681	-1.354360	51.437875	-1.700580	-5.780217	16.610880	-193.701384	-20.940131
Max resolution	1633.146553	570949.725569	491150.215456	1840.717836	-610.363161	13490.526141	-211.925195	-1636.741349	4811.488099	18249.019204	-1399.058778
Low resolution	1545.303242	491150.215456	517531.874048	1669.062486	-467.454111	14154.644868	46.353900	-1595.056176	4676.013450	10002.104040	-1768.925262
Effective pixels	5.380681	1840.717836	1669.062486	7.369343	-1.580111	46.475669	-1.798677	-5.702070	14.944016	58.073933	-2.747238
Zoom wide (W)	-1.354360	-610.363161	-467.454111	-1.580111	9.827211	-5.599643	3.382654	1.739377	-8.441360	-111.144747	-4.916978
Zoom tele (T)	51.437875	13490.526141	14154.644868	46.475669	-5.599643	7415.437800	-139.268478	-151.711765	142.885533	4980.931086	157.472876
Normal focus range	-1.700580	-211.925195	46.353900	-1.798677	3.382654	-139.268478	286.307952	33.514193	42.722384	-132.969248	-8.411771
Macro focus range	-5.780217	-1636.741349	-1595.056176	-5.702070	1.739377	-151.711765	33.514193	57.166422	-23.290299	-39.128418	7.573505
Storage included	16.610880	4811.488099	4676.013450	14.944016	-8.441360	142.885533	42.722384	-23.290299	710.631644	-146.695658	-31.278261
Weight (inc. batteries)	-193.701384	18249.019204	10002.104040	58.073933	-111.144747	4980.931086	-132.969248	-39.128418	-146.695658	65575.780224	3942.163493
Dimensions	-20.940131	-1399.058778	-1768.925262	-2.747238	-4.916978	157.472876	-8.411771	7.573505	-31.278261	3942.163493	435.625040
Price	-48.173224	105028.653377	73199.905112	411.308819	-230.516839	-718.435279	-747.190102	-79.603642	-379.230687	92118.719922	5312.905112

Does PCA help visualize the data? Do we get any insights from histograms/ bar charts/ line plots,etc.?



Logout

Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

+ %< > Run Code

Data Preprocessing:

KMeans Classification which is a clustering Algorithm:

In [19]: 1 from sklearn.cluster import KMeans
2 import sklearn.metrics as sm

In [20]: 1 chosen=['Release date','Max resolution','Low resolution','Effective pixels','Zoom wide (W)','Zoom tele (T)','Normal focus range']
2 X=df[chosen].values
3 model = KMeans(n_clusters=2)
4 model.fit(X)

Out[20]: KMeans(n_clusters=2)

In [23]: 1 from sklearn.preprocessing import StandardScaler

In [24]: 1 x = StandardScaler().fit_transform(X)
2 print(x)

```
[[-2.41993726e+00 -1.92394895e+00 -1.71235827e+00 ... 3.67757303e-01
-5.65358748e-01 -3.66253558e-01]
[-2.05275485e+00 -1.58498748e+00 -1.71235827e+00 ... 3.67757303e-01
2.45453669e+00 -3.66253558e-01]
[-1.31839003e+00 -2.43239115e+00 -3.77826576e-04 ... -2.22084248e-16
-3.80754166e-02 -3.66253558e-01]
...
[-9.51207616e-01 -5.68103081e-01 -1.17832051e+00 ... 5.52015101e-02
9.85943170e-03 -5.20183402e-01]
[-9.51207616e-01 -1.02031065e-01 -9.33553196e-01 ... 5.52015101e-02
9.85943170e-03 -5.20183402e-01]
[-5.84025204e-01 -1.16128565e+00 -1.48984254e+00 ... -5.69910075e-01
-9.96772382e-01 -4.32035542e-01]]
```

Edit View Insert Cell Kernel Widgets Help

PCA Analysis:

```
In [25]: 1 from sklearn.decomposition import PCA
2         pca = PCA(n_components=2)
3         principalComponents = pca.fit_transform(x)
4         principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
5         principalDf
```

```
Out[25]:
```

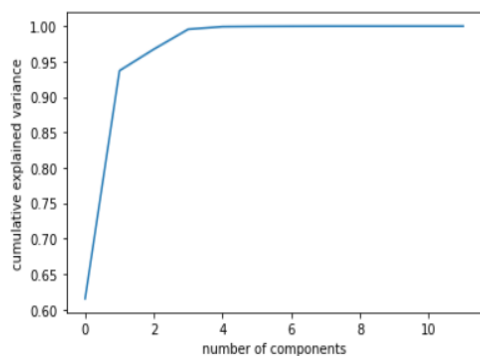
	principal component 1	principal component 2
0	-4.063260	-0.174233
1	-3.565766	1.610672
2	-2.576437	-0.571807
3	-2.759169	-0.010697
4	-2.878626	0.718695
...
1033	-1.795036	0.290815
1034	-1.893865	0.238052
1035	-1.785925	-0.206638
1036	-1.395752	-0.194060
1037	-2.749791	-1.382585

1038 rows x 2 columns

Edit View Insert Cell Kernel Widgets Help

Graphs of PCA Analysis:

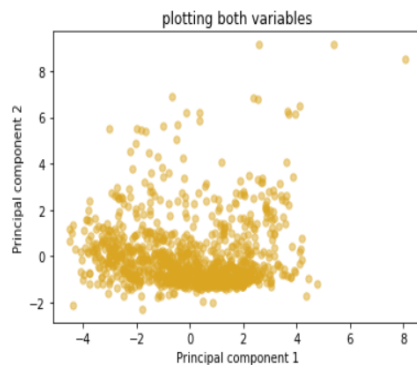
```
In [26]: 1 pca = PCA().fit(X)
2         plt.plot(np.cumsum(pca.explained_variance_ratio_))
3         plt.xlabel('number of components')
4         plt.ylabel('cumulative explained variance')
5         plt.show()
```



```
In [27]: 1 print(pca.explained_variance_ratio_)

[6.15080895e-01 3.21996547e-01 3.03836329e-02 2.81671753e-02
 3.68629908e-03 3.84216107e-04 1.61655928e-04 1.06458800e-04
 2.63548059e-05 5.13158929e-06 8.89201917e-07 7.45019327e-07]
```

```
In [28]: 1 plt.scatter(principalDf['principal component 1'],principalDf['principal component 2'],s=30,c='goldenrod',alpha=0.5)
2 plt.title('plotting both variables')
3 plt.xlabel('Principal component 1')
4 plt.ylabel('Principal component 2')
5 plt.show()
```

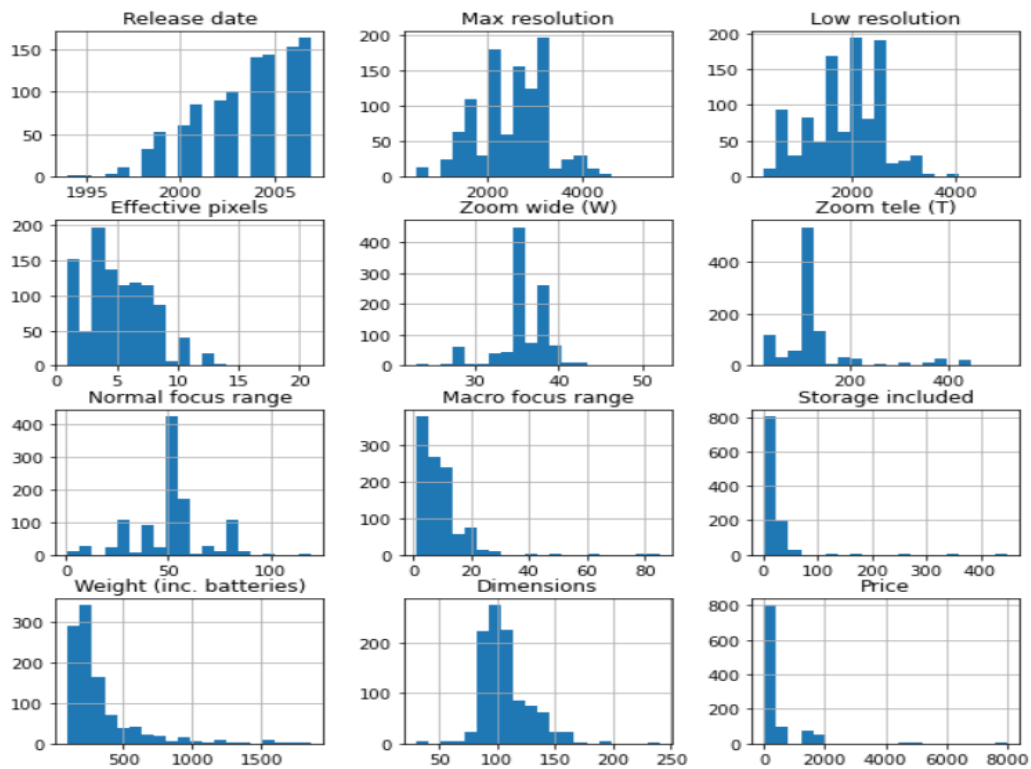


Data Visualization:

Histogram:

```
In [29]: 1 df.hist(column=["Release date","Max resolution","Low resolution","Effective pixels","Zoom wide (W)","Zoom tele (T)","Normal focus range","Macro focus range","Storage included","Weight (inc. batteries)","Dimensions","Price"], dtype=object)
```

```
Out[29]: array([[<AxesSubplot:title={'center':'Release date'}>,<AxesSubplot:title={'center':'Max resolution'}>,<AxesSubplot:title={'center':'Low resolution'}>],<AxesSubplot:title={'center':'Effective pixels'}>,<AxesSubplot:title={'center':'Zoom wide (W)'}>,<AxesSubplot:title={'center':'Zoom tele (T)'}>,<AxesSubplot:title={'center':'Normal focus range'}>,<AxesSubplot:title={'center':'Macro focus range'}>,<AxesSubplot:title={'center':'Storage included'}>,<AxesSubplot:title={'center':'Weight (inc. batteries)'}>,<AxesSubplot:title={'center':'Dimensions'}>,<AxesSubplot:title={'center':'Price'}>]], dtype=object)
```

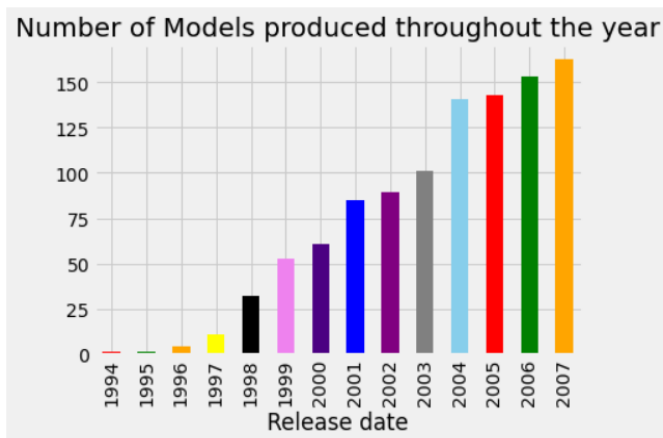


Bar Chart:

```
In [30]: 1 plt.style.use('fivethirtyeight')
2 print(df.pivot_table(index=['Release date'], aggfunc='size').plot(kind='bar', color=['red', 'green', 'orange', 'yellow', 'black'],
3 plt.ylabel=["No. of cameras"]
4 plt.title("Number of Models produced throughout the year"))
```

AxesSubplot(0.08,0.07;0.87x0.81)

Out[30]: Text(0.5, 1.0, 'Number of Models produced throughout the year')



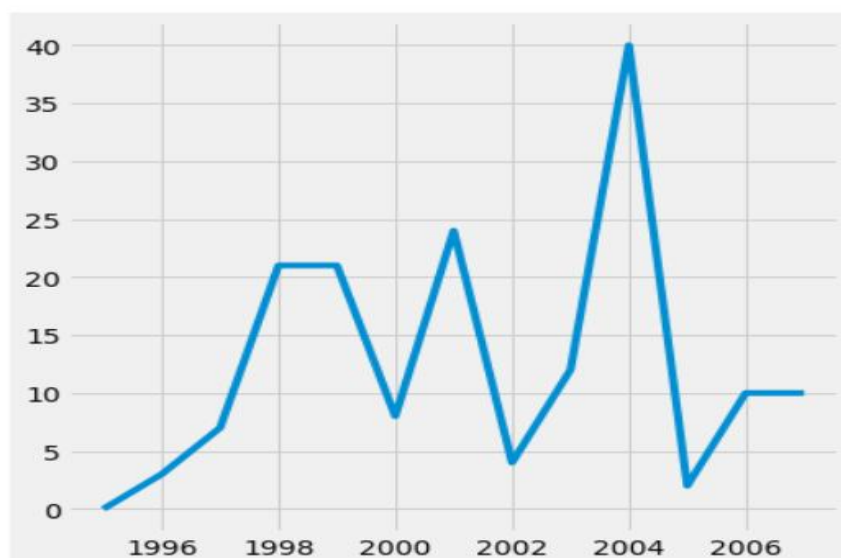
Edit View Insert Cell Kernel Widgets Help

+

Line Graph: Growth in Production

```
In [34]: 1 growth_rate=[]
2 growth=list(df["Release date"].value_counts())
3 dates=set(df["Release date"])
4 j=0
5 for i in growth:
6     growth_rate.append(int(j-i))
7     j=i
8 del(growth_rate[0])
9 growth.clear()
10 for i in growth_rate:
11     growth.insert(0,i)
12 del(growth_rate)
13 dates=list(dates)
14 del(dates[0])
15 plt.figure(figsize=(6,6))
16 plt.plot(dates,growth)
```

Out[34]: [<matplotlib.lines.Line2D at 0x29e6af658b0>]



Edit View Insert Cell Kernel Widgets Help

+

Plot representing no.of models manufactured by individual manufacturers:

```
In [35]: 1 Manufacturer=list()
2 release_dates=set(df["Release date"])
3 for i in df['Model']:
4     if('Agfa' in i):
5         Manufacturer.append('Agfa')
6     elif('Canon' in i):
7         Manufacturer.append('Canon')
8     elif('Casio' in i):
9         Manufacturer.append('Casio')
10    elif('Contax' in i):
11        Manufacturer.append('Contax')
12    elif('Epson' in i):
13        Manufacturer.append('Epson')
14    elif('Fujifilm' in i):
15        Manufacturer.append('Fujifilm')
```

```


16 elif('HP' in i):
17     Manufacturer.append('HP')
18 elif('JVC' in i):
19     Manufacturer.append('JVC')
20 elif('Kodak' in i):
21     Manufacturer.append('Kodak')
22 elif('Kyocera' in i):
23     Manufacturer.append('Kyocera')
24 elif('Leica' in i):
25     Manufacturer.append('Leica')
26 elif('Nikon' in i):
27     Manufacturer.append('Nikon')
28 elif('Olympus' in i):
29     Manufacturer.append('Olympus')
30 elif('Panasonic' in i):
31     Manufacturer.append('Panasonic')
32 elif('Pentax' in i):
33     Manufacturer.append('Pentax')
34 elif('Ricoh' in i):
35     Manufacturer.append('Ricoh')
36 elif('Samsung' in i):
37     Manufacturer.append('Samsung')
38 elif('Sanyo' in i):
39     Manufacturer.append('Sanyo')
40 elif('Sigma' in i):
41     Manufacturer.append('Sigma')
42 elif('Sony' in i):
43     Manufacturer.append('Sony')
44 elif('Toshiba' in i):
45     Manufacturer.append('Toshiba')
46 df['Manufacturer']=Manufacturer

```


```

In [36]: 1 df.pivot_table(index=['Manufacturer'], aggfunc='size').plot(kind='barh',color=['red','green','orange','yellow','black','viol

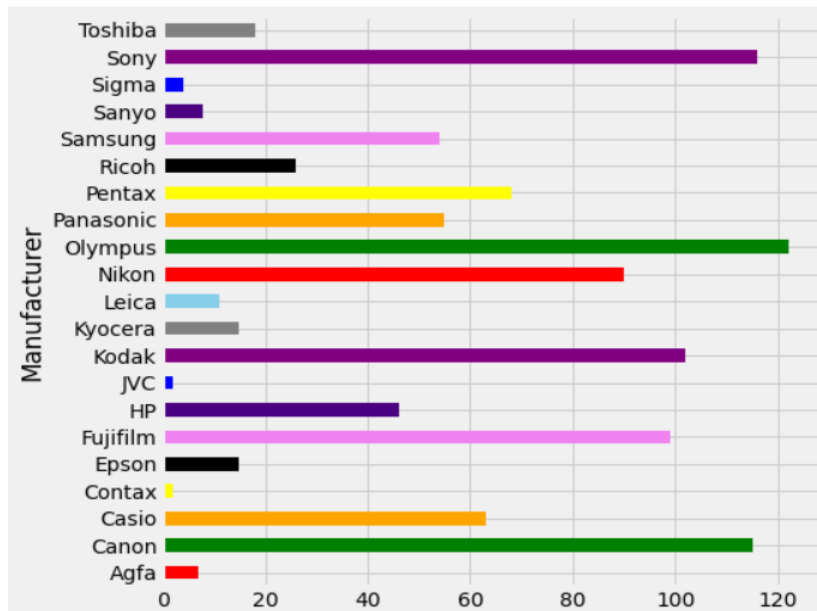
```

 **Jupyter** Team Data Pirates EDA and Data Visualization Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

       Run    Markdown 

Out[36]: <AxesSubplot:ylabel='Manufacturer'>



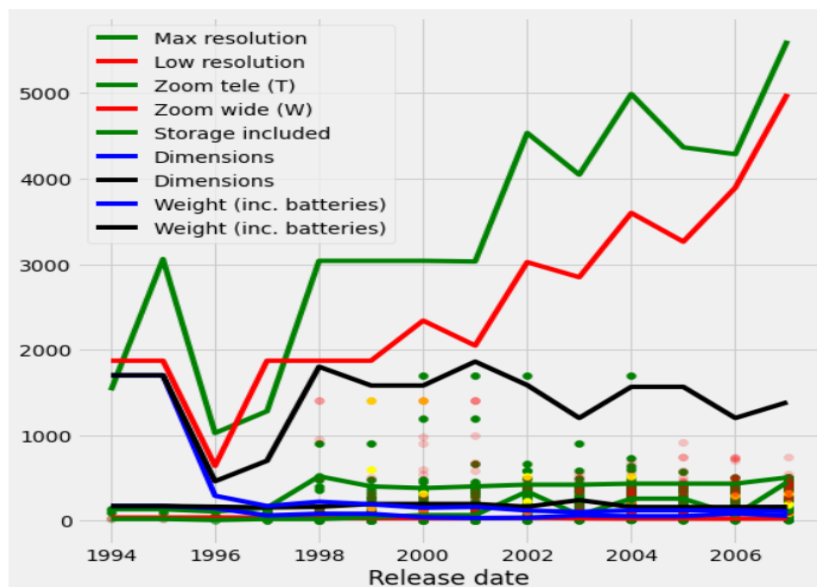
Combination of Line and Scatter plot: Showing the trend of the features of Dataset

```
In [43]: 1 g_df=df.groupby('Release date')

In [44]: 1 df['Range'] = (df['Zoom tele (T)'] - df['Zoom wide (W)']).abs()

In [45]: 1 plt.figure(figsize=(8,8))
2 plt.scatter(x=df['Release date'], y=df['Zoom tele (T)'],c='green')
3 plt.scatter(x=df['Release date'], y=df['Zoom wide (W)'],c='red',alpha=0.4)
4 plt.scatter(x=df[df['Range'] >25]['Release date'], y=df[df['Range'] >25]['Range'],c='green')
5 plt.scatter(x=df[(df['Range'] >100) & (df['Range'] < 200)]['Release date'], y=df[(df['Range'] >100) & (df['Range'] < 200)]['Range'],c='green')
6 plt.scatter(x=df[(df['Range'] >200) & (df['Range'] < 250)]['Release date'], y=df[(df['Range'] >200) & (df['Range'] < 250)]['Range'],c='green')
7 plt.scatter(x=df[df['Range'] >250]['Release date'], y=df[df['Range'] >250]['Price'],c='Red', alpha=0.2)
8 plt.scatter(x=df['Release date'], y=df['Storage included'],c='green',alpha=0.2)
9 plt.scatter(x=df[(df['Range'] >100) & (df['Range'] < 200)]['Release date'], y=df[(df['Range'] >100) & (df['Range'] < 200)]['Dimensions'],c='Red', alpha=0.2)
10 plt.scatter(x=df[(df['Range'] >200) & (df['Range'] < 250)]['Release date'], y=df[(df['Range'] >200) & (df['Range'] < 250)]['Dimensions'],c='Red', alpha=0.2)
11 plt.scatter(x=df[df['Range'] >250]['Release date'], y=df[df['Range'] >250]['Dimensions'],c='Red', alpha=0.2)
12 plt.scatter(x=df[(df['Range'] >100) & (df['Range'] < 200)]['Release date'], y=df[(df['Range'] >100) & (df['Range'] < 200)]['Weight (inc. batteries)'],c='Red', alpha=0.2)
13 plt.scatter(x=df[(df['Range'] >200) & (df['Range'] < 250)]['Release date'], y=df[(df['Range'] >200) & (df['Range'] < 250)]['Weight (inc. batteries)'],c='Red', alpha=0.2)
14 plt.scatter(x=df[df['Range'] >250]['Release date'], y=df[df['Range'] >250]['Weight (inc. batteries)'],c='Red', alpha=0.2)
15 g_df['Max resolution'].max().plot(color='green',legend=True)
16 g_df['Low resolution'].max().plot(color='red',legend=True)
17 g_df['Zoom tele (T)'].max().plot(color='green',legend=True)
18 g_df['Zoom wide (W)'].min().plot(color='red',legend=True)
19 #g_df['Range'].max().plot(color='green',legend=True)
20 g_df['Storage included'].max().plot(color='green',legend=True)
21 g_df['Dimensions'].min().plot(color='blue',legend=True)
22 g_df['Dimensions'].max().plot(color='black',legend=True)
23 g_df['Weight (inc. batteries)'].min().plot(color='blue',legend=True)
24 g_df['Weight (inc. batteries)'].max().plot(color='black',legend=True)
25
```

Edit View Insert Cell Kernel Widgets Help Out[45]: <AxesSubplot:xlabel='Release date'>



Link for Detail EDA and Data Visualization: <https://github.com/Pawantech-App/DATA-ANALYTICS-PROJECT-/blob/main/Team%20Data%20Pirates%20EDA%20and%20Data%20Visualization.ipynb>

Literature Survey:

Link for google sheet:

<https://docs.google.com/spreadsheets/d/1v3aQludyhH9dT-PMYH3ol4m8WgamyRfT5raQEjHoksM/edit#gid=0>

Literature Survey Summary:

Paper: Tracking of Humans and Estimation of Body/Head Orientation from Top-view Single Camera for Visual Focus of Attention Analysis.

Summary: In this paper the author has used a Top-view Single Camera for Visual Focus of Attention Analysis for Tracking Humans and Estimation of Body/Head Orientation. Analysis of human behavior in public places is an important topic which has attracted much attention from many researchers, designers, companies and organizations and the author has tried to work on this field. This Paper addresses the problem of determining a person's body and head orientations while tracking the person in an indoor environment monitored by a single top-view camera. The challenging part of this problem lies in the wide range of human postures depending on the position of the camera and articulations of the pose. The author has used two-level cascaded particle filter and optical flow of SIFT (scale-invariant feature transform) approach in that SOFV (head region) is introduced to track humans. The two levels are Color clues are used as the first level for each iteration and edge-orientation histograms are reutilized to support the tracking at the second level. To determine the body and head orientations, a combination of Shape Context and SIFT features is proposed. The author has used the features of camera like its focal length, lens, zoom, etc. The author has used various approaches like one which is mentioned already and also various algorithm like Human tracking algorithm and Human Body/Head Estimation. It is composed of three main parts that are Motion Vector Detection, SIFT Optical Flow, Body Orientation Check. The author also worked on computer vision and also many other projects. The conclusion is that a new approach has been found by combining shape context matching and optical flow of SIFT features is proposed to detect the orientation change in the body and head. The system presented in this paper is an initial part of a project, which focuses on wandering people while walking or standing still. The proposed system will be improved in future to apply to more complex situations in public places.

Paper: Selecting the Optimal Focus Measure for Autofocusing and Depth-From-Focus

Summary: In this paper the author has proposed a approach in improving the Autofocusing and Depth-From-Focus by selecting the Optimal Focus Measure. In this paper the author has used various approaches are used. A method is described for selecting the optimal focus measure with respect to gray-level noise from a given set of focus measures in passive autofocusing and depth-from-focus applications. The method is based on two new metrics that have been defined for estimating the noise-sensitivity of different focus measures. The first metric—the Autofocusing Uncertainty Measure (AUM)—is useful in understanding the relation between gray-level noise and the resulting error in lens position for autofocusing. The second metric—Autofocusing Root-Mean-Square Error (ARMS error)—is an improved metric closely related to AUM. AUM and ARMS error metrics are based on a theoretical noise sensitivity analysis of focus measures, and they are related by a monotonic expression. The theoretical result and experimental result are used to find the accuracy of the model. The various approaches are AUTOFOCUSING ALGORITHM, AUTOFOCUSING UNCERTAINTY MEASURE (AUM), ARMS ERROR. The conclusion of this paper is divided into First, explicit expressions for the variance of other focus measures such as sum of absolute values of image derivatives could be derived so that ARMS error can be used to estimate their autofocusing accuracy. Second, in the definition of ARMS error, the local smoothness of focus measures could be modeled differently than here. Third, deriving an optimal focus measure filter for a given image and noise level remains to be investigated. ARMS error has been defined as a metric for selecting the optimal focus measure for autofocusing with respect to gray-level noise from a given set of focus measures. It is based on the assumption of local smoothness of focus measures with respect to lens position. ARMS error can be applied to any focus measure whose variance can be expressed explicitly as a function of gray-level noise variance.

Paper: Camera calibration and precision analysis based on BP neural network*

Summary: In this Paper the author has used the new approach of BP Neural Network which a new approach based on BP neural network forseeking-solution projection matrix of camera is presented in this hypothesis. The sum of square of errors between the network outputs and constant vector is taken as performance index. The weight matrix between the input layer and hidden layer is tuned in the light of gradient descend method and can be achieved stable value while the performance index is reached to expected value, while two elements of weight matrix between hidden and output layer are coordinates of projected points in image plane and keep constant during the training. Thus the projection matrix of cameras can be obtained from the weights between input layer and hidden layer of the neural network, and calibration of system is finished. Finally, the precision analysis is carried out for machine vision system. The calibration of camera is based on the BP Neural Network and also precision analysis is done by finding the errors between coordinates in the actual image and coordinates of projected points obtained with projection matrix and corresponding 3D coordinates. The conclusion of this paper is the projection matrix of camera is obtained from stable weight vectors between input layer and hidden layer of the network in the experiment. In precision analysis, the coordinates of projected points in the image plane are obtained from the stable weight matrix between the hidden layer and output layer of the net in the light of gradient descent method.

Paper: User-based design specifications for the ultimate camera trap for wildlife research

Summary:

The adoption of camera trapping in place of traditional wildlife survey methods has become common despite inherent flaws in equipment and a dearth of research to test their fit for purpose. We investigated the demand for white-flash camera traps around the world to highlight the demand for such camera traps in wildlife research to the manufacturing industry. We also compiled the camera-trap specifications required by scientists through the world in an effort to influence and improve the quality of camera traps for research. We carried out an internet-based survey of biologists, zoologists, conservationists and other wildlife researchers by using a questionnaire to gather baseline market data on camera-trap use and demand. We also conducted an informal survey of scientists via email and in person, asking for their preferences and features of an ultimate camera-trap design. The results of our surveys and analysis have brought about the conclusions that, The Infrared camera traps are widely used and more so than white-flash camera traps, although the demand for white flash remains significant. Cost, speed, size, ease of use, versatility and the range of settings were the key features identified in a good camera trap. From the above points of discussion we come across the final conclusions that, The present paper describes and discusses the desired features and specifications as defined by over 150 scientists using camera traps around the world. Data gathered also provide some insight into the market demand for camera traps by biologists, zoologists, conservationists and other wildlife researchers around the world.

Paper: Very high-resolution imaging scheme with multiple different-aperture cameras

Summary: Towards the development of a very high definition (VHD) image acquisition system, previously we developed the signal processing based approach with multiple cameras. This approach produces an improved resolution image with sufficiently high signal-to-noise ratio by processing and integrating multiple images taken simultaneously with multiple cameras. Originally, in this approach, we used multiple cameras with the same pixel aperture, but in this case there are severe limitations both in the arrangement of multiple cameras and in the configuration of the scene, in order to guarantee the spatial uniformity of the resultant resolution. To overcome this difficulty completely, this work presents the utilization of multiple cameras with different pixel apertures, and develops a new, alternately iterative signal processing algorithm available in the different aperture case. Experimental simulations clearly show that the utilization of multiple different-aperture cameras prospects to be good and that the alternately iterative algorithm behaves satisfactorily.

Paper: Requirements for Camera Calibration: Must Accuracy Come with a High Price?

Summary: Since a large number of vision applications rely on the mapping between 3D scenes and their corresponding 2D camera images, an important practical consideration for researchers is, what are the major determinants of camera calibration accuracy and what accuracy can be achieved within the practical limits of their environments. In response, we present a thorough study investigating the effects of training data quantity, measurement error, pixel coordinate noise, and the choice of camera model, on camera calibration results. Through this effort, we seek to determine whether expensive, elaborate setups are necessary, or indeed, beneficial, to camera calibration, and whether a high complexity camera model leads to improved accuracy. The results are first provided for a simulated camera system and then verified through carefully controlled experiments using real-world measurements.

Paper: Smart Camera for Quality Inspection and Grading of Food Products

Summary: With the growth in the food related industry most countries including United States have mechanized all the process involved. With the functions of the machines differing in numerous ways, the technology by humans has managed to remove all these problems and made the machines were adaptable to a wide range of tasks, without human intervention. Two data sets were created as examples of quality evaluation for speciality agriculture, Date skin quality evaluation and oyster shape quality. For better results I had took only one data set i.e, date skin quality evaluation. Methods includes 1. Visual inspection algorithm

2. Feature composition
3. Evolutionary learning
4. Feature selection
5. Smart camera

Original image was 200*800, it was cropped and collected 200*300 images. Images were captured on the blue plastic chains, it was very bright in the IR image, delamination or dry date skin is slightly darker and moist date is darkest.

Vision technology is proven solution for food grading and inspection since the late 1980s. Sophisticated and more powerful machines deep learning method which easily perform same visual inspection tasks with impressive result. Unlike other vision applications, it operates in indoor under LED lights and regulated voltage.

Paper: Camera-Based Blind Spot Detection with a General Purpose Lightweight Neural Network

Summary: Blind spot detection is an important feature of Advanced Driver Assistance Systems (ADAS). The paper provides camera-based deep learning method that accurately detects other vehicles in the blind spot, replacing the traditional higher cost solution using radars. Blind spot detection, as a real-time embedded system application, requires high speed processing and low computational complexity. Hereby, they proposed a novel method that transfers blind spot detection to an image classification task. Goal is to design a deep neural network model that can solve real world problems without involving too many layers. The basic principle to design a lightweight neural network is reducing the model size without losing too much accuracy. Data set included is the CIFAR-10 dataset which is a popular dataset for evaluation of machine learning methods. It contains 60,000 32×32 color images in 10 different categories. The model is classified as a Car or No-Car two-class classification problem. For each dataset, we use the same learning rate and batch size to make a fair comparison among four different neural networks. Therefore, the combining of depth-wise separable convolution, residual learning and squeeze-and-excitation is the best tradeoff between accuracy and cost. The major drawback is they haven't yet considered all the situations with different roads and weather conditions due to the additional workload of gathering and labeling data. At last authors declare no conflict of interest.

Paper: Real-Time Human Action Recognition with a Low-Cost RGB Camera and Mobile Robot Platform

Summary: As the aging population and one-person households increase worldwide, societal and political aspects affect health care. Human action recognition is an important research area in the field of computer vision that can be applied in surveillance, assisted living, and robotic systems interacting with people. Although various approaches have been widely used, recent studies have mainly focused on deep-learning networks using Kinect camera that can easily generate data on skeleton joints using depth data, and have achieved satisfactory performances. However, their models are deep and complex to achieve a higher recognition score; therefore, they cannot be applied to a mobile robot platform using a Kinect camera. To overcome these limitations, we suggest a method to classify human actions in real-time using a single RGB camera, which can be applied to the mobile robot platform as well. The deep learning technique has exhibited superior performances in image recognition, speech recognition, natural language processing, and human action recognition. In this section, at first, they have introduced the method of extracting a 3D skeleton joint using a single RGB camera instead of a Kinect camera, and later, have described the converting process of skeleton data to image form for using it in the embedded system. Afterward, a simple CNN model architecture has been introduced. Data set includes: The proposed method was evaluated on the NTU-RGBD dataset, which is the largest dataset collected by the Kinect V2 camera. The most challenging part of this dataset is that it is typically recorded on a variant view that mainly covers three different views ($-45, 0, 45$). The dataset has four modalities, depth maps, RGB frames, and IR sequences. In addition, the results of the proposed method showed high accuracy for distinct actions such as standing, sitting, hopping, and jumping. However, other actions, in particular, that mainly used the arm joints, showed less accuracy than the Kinect data. Therefore, proposed method using the RGB camera achieved an accuracy of 71%, and Kinect-17 and Kinect-25 achieved accuracies of 74% and 75%, respectively. In their opinion, more joint information and precise joint estimation correlate with accuracy. All the processes of joint extraction and action recognition were conducted on

an average of 15 FPS on the Xavier board and was applied to a mobile robot platform to use as a monitoring system in the in-doors. Finally, they presented a human tracking algorithm. With this method, they suggest that more variable action labels can be used by utilizing RGB images via formerly recorded videos that were performed in a real-time embedded system. For future work, optimized networks are needed for higher performances, so that they can be used in real-time within the embedded board. The main limitation of this method that recognizing the front view of the person, a more precise pose estimation algorithm with robustness has to be developed for self-occlusion problems. At last authors declare no conflict of interest.

Paper: Price Effects in Online Product Reviews: An Analytical Model and Empirical Analysis

Summary: Consumer reviews may reflect not only perceived quality but also the difference between quality and price (perceived value). In camera markets where product prices change frequently, this price influenced reviews may be biased as a signal of product quality when used by consumers possessing no knowledge of historical prices. In this paper, they developed an analytical model that examines the impact of price-influenced reviews on firm optimal pricing and consumer welfare. They quantified the price effects in consumer reviews for different formats of review systems using actual market prices and online consumer ratings data collected for the digital camera market. The empirical results suggest that unidimensional ratings, commonly used in most review systems, can be substantially biased by price effects. In fact, unidimensional ratings are more closely correlated with ratings of product value than ratings of product quality. The findings suggest the importance for firms to account for these price effects in their overall marketing strategy and suggest that review systems could better serve consumers by explicitly expanding review dimensions to separate perceived value and perceived quality. Firms may benefit by reducing their prices to boost their ratings in systems that cannot distinguish price from quality, although the existence of these price effects does not always benefit firms in that they may be worse off in a scenario in which this type of review bias is present.

Paper: The Effects of Camera Resolution and Distance on Suspect Height Analysis Using Photo Modeler

Summary: The purpose of this study was to examine how camera resolution and suspect-camera distance affect the accuracy and precision of suspect height estimations using Photo Modeler software. Sixteen individuals were measured and recorded standing at 15 pre-set distances on 7 security cameras, each with a different resolution setting. A height scale was used to measure each individual's height prior to recording and was also used as a reference height. Height estimates were taken in Photo Modeler by extracting video frames that were calibrated using 3D point model data obtained from a laser scan of the test site. Errors were calculated for the measurements and compared using the Kruskal-Wallis H-test, which indicated significant differences for errors among different resolutions and distances ($p < 0.01$). Interaction plots, however, demonstrated little difference in errors for most resolutions and positions. The accuracy and precision of height estimates began to decrease with resolutions under 960H and distances over 36.5 m. The purpose of this research was to examine if camera resolution and suspect-camera distance affect the accuracy and precision of photogrammetric height measurements. This study was performed under optimal conditions where multiple variables that can be encountered in the field were controlled for..

That considered, the majority of measurement errors (75%) were below +/- 1.3 cm. This result, and the maximum result, can act as a guideline for analysts and investigators when performing and considering the reliability of suspect height analysis for supporting evidence. As the data suggests that camera-to-head angle may affect accuracy and precision at particular distances, this variable requires further study

Paper: First Steps Towards Camera Model Identification with Convolutional Neural Networks

Summary: Detecting the camera model used to shoot a picture enables to solve a wide series of forensic problems, from copyright infringement to ownership attribution. For this reason, the forensic community has developed a set of camera model identification algorithms that exploit characteristic traces left on acquired images by the processing pipelines specific of each camera model. In this paper, they investigated a novel approach to solve camera model identification problem. They proposed a data-driven algorithm based on convolutional neural networks, which learns features characterizing each camera model directly from the acquired pictures. Later the selected CNN is used extract feature vector of 128 elements which is then used in SVM Training. In order to validate the proposed algorithm, they performed a set of experiments in different operative scenarios:

- 1) They compared the algorithm against recent state-of-the-art methods;
- 2) They evaluated the generalization capability of the proposed CNN.

This proposed method outperforms up-to-date state-of-the-art algorithms on classification of 64×64 color image patches. The features learned by the proposed network generalize to camera models never used for training. The overall accuracy reaches 93%, showing that the features extracted with the CNN Model, trained on 18 camera models from the Dresden Image Dataset, are capable of generalizing to unknown camera models.

Your Plan:

Our project objective is to predict the camera model based on the given features and price of the camera. We will use the machine learning algorithms of KNN(k-nearest neighbours), Logistic Regression and SVM.

References:

PAWAN PRASAD

- 1) <https://ieeexplore.ieee.org/abstract/document/5457590/>
- 2) <https://ieeexplore.ieee.org/abstract/document/709612/>
- 3) <https://ieeexplore.ieee.org/abstract/document/5305383/>

PHANI KUMAR VEDURUMUDI

- 1) <https://www.publish.csiro.au/wr/wr12138>

- 2) <https://www.sciencedirect.com/science/article/pii/092359659390014K>
- 3) <https://ieeexplore.ieee.org/abstract/document/4129503/>

G U DEEPAK

- 1) <https://www.mdpi.com/668458>
- 2) <https://www.mdpi.com/414372>
- 3) <https://www.mdpi.com/720690>

JAYANTH O

- 1) <https://www.jstor.org/stable/25750706>
- 2) <https://www.sciencedirect.com/science/article/pii/S0379073820304631>
- 3) <https://ieeexplore.ieee.org/abstract/document/7786852/>

Project Details:

<https://github.com/Pawantech-App/DATA-ANALYTICS-PROJECT-/blob/main/Team%20Data%20Pirates%20EDA%20and%20Data%20Visualization.ipynb>