EDA HandWritten Notes

- Darshan R M

"Learn, Share, and Collaborate"

Week - 14

Exploratory data analysis

EDA

→ is done to understand specific dataset.

Wine

* summary → df.info()
* descriptive statistics → df.describe()
* list down column name → df.columns
* shape → df.shape
* df['—'].unique() → unique/possible value
* null values → df.isnull().sum()
* duplicate values ⇒ df.duplicated()
  record
  ↓ drop
  df.drop_duplicates()
* sns.heatmap(df.corr()) → Correlation
  ↓
  annot=True

* df['—'].value_counts()

↓plot

> df['—'].value_counts().plot(kind = 'bar')

* pairplot

↳ for univariate, bivariate analysis

> sns.pairplot(df)

↓

To see how 1 feature is related to other and that feature with other feature.

* sns.catplot(x='—', y='—', data=df, kind='scatter')

---

* Data checks

Write insights for each

*① check missing values.
② check duplicates.
③ check data type.
④ check the number of unique values of each column.
⑤ check statistics of data set.
⑥ check various categories present in different categorical columns.

* missing values → (feature engineering) replace, drop based on amount of data.

* duplicate value → should be removed as ML need different kind of data.

* `df.nunique()` → Each column unique values

---

* separate numerical & categorical feature

Categor = [fea for fea in df.columns if df[fea].dtype == 'object']

numerical → 'int64'

---

fig, axis = plt.subplots(1, 2, figsize=(-, -))
row, column

→ plt.subplot(1,2)
1st row 1st column 1st figure

→ sns.histplot(data=df, x='average', bins=30, kde=True, color='g')
hue = 'gender'
↓
based on gender

(male)

sns.histplot(data = df[df['gender']=='male'], x='average', kde=True, hue='lunch')

↳ 'female'

df = pd.read("—", header=1)

↳ making first row as header.

Forest Fire

① load dataset

② df.info()

*datatype not correct to fix it no string should be there

③ remove string and nan values.

> df[df.isnull().any(axis=1)]

creating region column

df.loc[:122, 'Region'] = 1

df['region'] = df['Region'].astype(int)

~~df.isnull().~~

df = df.dropna().reset_index(drop=True)

drop(122), — " —

④ removing extra space present in column name

→ df.columns.str.strip()

(objects)

⑤ change datatype to corresponding datatype.

⑥ df.describe()

⑦ removing outliers

→ Storing cleaned data in new file.

EDA → Encoding : df['cl~'] = np.where(df['class'] == 'not fire', 0, 1)

density plot for all features

↓

```
plt.style.use('seaborn')
df.hist(bins=30, figsize=(10,15), ec='b')
plt.show()
```

Flight

* Date
  12/6/2023 ⇒ need to convert into separate date month & year Column

object
df['date'] = df['date_of_jou'].str.split('/').str[0]
objecto
0th index

month [1]
year [2]

→ convod into numerical
↓ typecast
df['Date'] = df['Date'].astype(int)

→ drop
df.drop('—', axis=1, inplace=True)
Column

arrival time
21:35 to march

> df['arrival_hour'] = df['—'].str.split(':').str[0]
> df['arrival_min'] = df['—'].str.split(':')[1]

object
d['arr_min'] = df['arr_min'].str.split('—').str[0]
min 35 10 march

integer
df['—'] = df['—'].astype(int) → drop

10:32

* Similarly, "dep_Time" feature

* Total_stops - Categorical

> df['---'].unique

> df['--'] = df['--'].map({'non-stop': 0, 'one stop': 1, ---})

Convert into ordinal label

np.nan : 1

* Duration

"2h 50m"

> df['hour'] = df['duration'].str.split(' ').str[0].str.split('h').str[0]

* same with min - 1 [← .str[1]
('m').str[0]

* df.at[---, '---'] = 68

* df.loc[---] ⇒ only need cannot waste

fixing nan values

* df['---'] = df['---'].replace(np.nan, 0)

(to_replace = ---, value = ---)

[-,-]

Categorical values

↳ OneHotEncoder encoding

> from sklearn.preprocessing import OneHotEncoder

> encoder = OneHotEncoder()

> encoded_df = pd.DataFrame(
encoder.fit_transform(df[['—', '—']]).toarray(),
columns = encoder.get_feature_names_out())

>

> Herald f = pd.Concat(encoded_df,

axis = 1)

Playstore

Playstore

df['—'].str.isnumeric().sum()

> df[~]

not

> df = df.drop(df.index[—])

record

```
for spec_char in char_to_rem:
    for col in col_to_clean:
        df[col] = df[col].str.replace(spec, '')
```

> January 7, 2018 ⟶ 2018-01-07

```
df['---'] = pd.to_datetime(df['---'])
```

> 2018-01-07

↓

```
df['day'] = df['last update'].dt.day
          .month
          .year
```

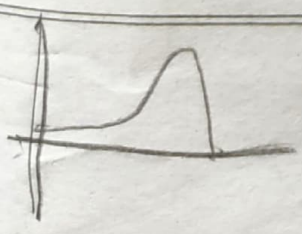
```
> df[col].value_counts()
    # on numerical values
df[col].value_counts(normalize=True)*100
    # on percentage
```

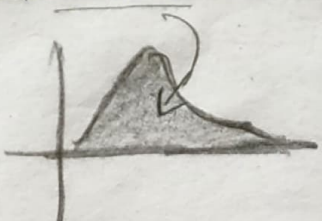
```
plt.suptitle(⟶)
```

Numerical

Kernel density estimator

```
> sns.kdeplot(x = df[col], shade = True)
                shade = True, color='_')
```





Categorical

* sns.countplot(x = df[—], palette='Set2')

---

pie

> df[—].value_counts().plot.pie(y = df[—], figsize=(15,6))

Top 10 app Categories

c = pd.DataFrame(df['Category'].value_counts())

> c.rename(columns={'Category': 'Count'}, inplace=True)

↓

---

bar plot

> sns.barplot(data = df[:10], x = df, index[:10], y = 'Count')

Q7 largest no of installation category

```
> dow = pd.DataFrame( df.groupby('category')['install'].sum())
> dow.sort_values( by = 'installs', ascending = False,
                   inplace = True)

> dow.reset_index(inplace = True)
```

```
sns.barplot( data = dow,
     x = dow['category'] [:10]
     y = dow['installs'] [:10])
```

---

index	value
0	'Hi'
1	'bye'
2	'20'

---

* Revision

```
① > from operator import concat
    from funtools import reduce
  > flat list = reduce ( concat, nested_list)
```

```
② d1 = { 'A': 20 --- }    |  d3 = { ** d1 , ** d2 }
   d2 = { 'x1': 100 --- } |        ↓
                          { 'A': 20, 'x1': 100, ---- }
```

unpacking operator