1. Problem and Data Description

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Spaceship Titanic Section

```
# Reading
#from google.colab import drive
#drive.mount('/content/drive')
ss_train = pd.read_csv('/content/ss_train.csv') ## this might be on the directory
ss_test = pd.read_csv('/content/ss_test.csv')
# Shape and preview
#print('Train set shape:', ss_train.shape)
#print('Test set shape:', ss_train.shape)
#ss_train.head()
#! ls 'content/drive/My Drive/ss_train.csv'
```

Housing Prices Section

```
hp_train = pd.read_csv('/content/hp_train.csv') ## this might be on the directory
hp_test = pd.read_csv('/content/hp_test.csv')
# Shape and preview
print('Train set shape:', hp_train.shape)
print('Test set shape:', hp_test.shape)
hp_train.head()
```

```
Train set shape: (1460, 81)
Test set shape: (1459, 80)
```

```
   Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street  Alley  LotShape  LandContour
```

2. Data Preprocessing & Exploratory Data Analysis

Spaceship Titanic Section

2.1 Handing missing values

```
## missing value check
print('Missing values from train set:')
print(ss_train.isna().sum())
print('Missign values from test set:')
print(ss_test.isna().sum())
```

```
Missing values from train set:
PassengerId       0
HomePlanet      201
CryoSleep       217
Cabin           199
Destination     182
Age             179
VIP             203
RoomService     181
FoodCourt       183
ShoppingMall    208
Spa             183
VRDeck          188
Name            200
Transported       0
dtype: int64
Missign values from test set:
PassengerId       0
HomePlanet       87
CryoSleep        93
Cabin           100
Destination      92
Age              91
VIP              93
RoomService      82
FoodCourt       106
ShoppingMall     98
Spa             101
VRDeck           80
```

```
Name            94
dtype: int64
```

```
ss_test.head(3)
```

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	Foo
0	0013_01	Earth	True	G/3/S	TRAPPIST-1e	27.0	False	0.0	
1	0018_01	Earth	False	F/4/S	TRAPPIST-1e	19.0	False	0.0	

```
ss_train.head(3)
```

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	Foo
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	

```
##spaceship trainset discription
ss_train.describe()
```

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	38.000000	47.000000	76.000000	27.000000	59.000000	46.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

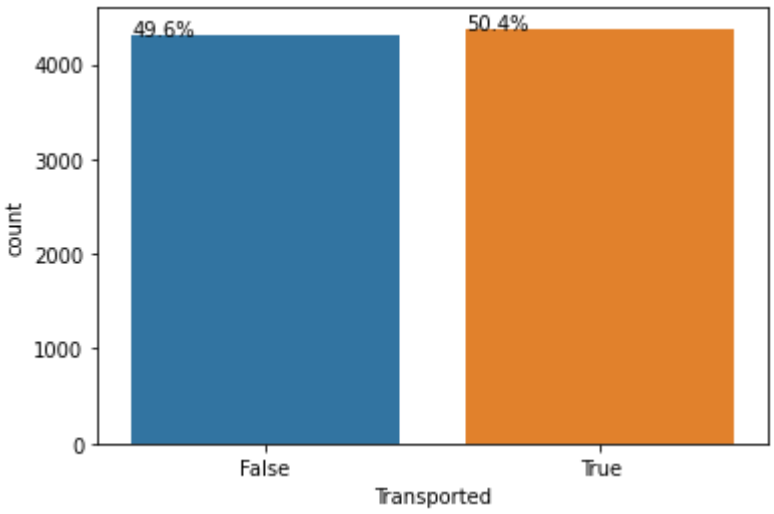
```
## spaceship test set discription
ss_test.describe()
```

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	4186.000000	4195.000000	4171.000000	4179.000000	4176.000000	4197.000000
mean	28.658146	219.266269	439.484296	177.295525	303.052443	310.710031
std	14.179072	607.011289	1527.663045	560.821123	1117.186015	1246.994742
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	26.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	37.000000	53.000000	78.000000	33.000000	50.000000	36.000000
max	79.000000	11567.000000	25273.000000	8292.000000	19844.000000	22272.000000

Fromt this we know taht there will be 6 numerical columns from the train and test datasets.

For the total sum graph transported or not

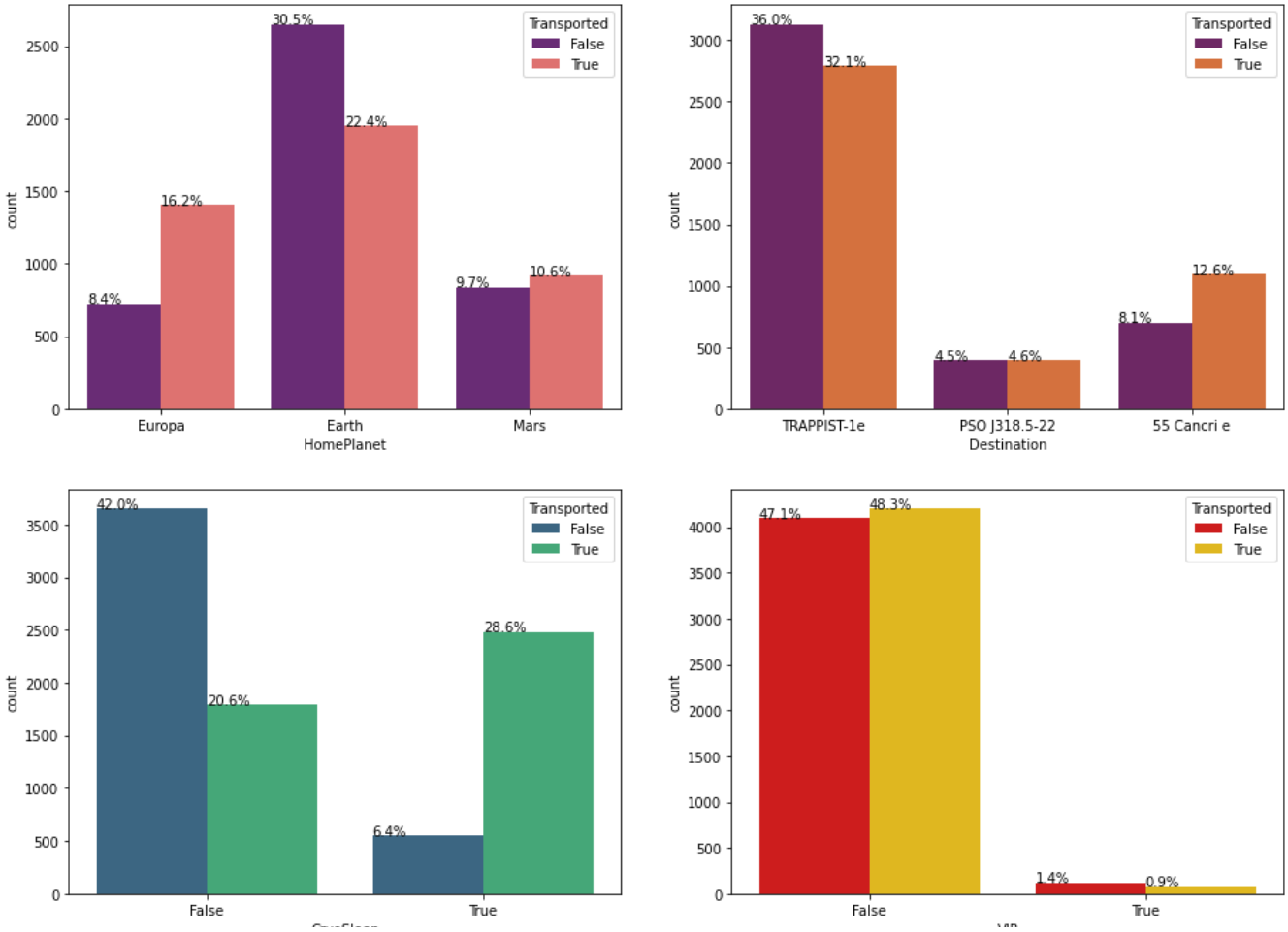
```
## use countplot to see in bar spot
total=float(len(ss_train['Transported']))
t=sns.countplot(data=ss_train, x='Transported')
for p in t.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/total)
    x = p.get_x()
    y = p.get_height()
    t.annotate(percentage, (x,y))
plt.show()
```



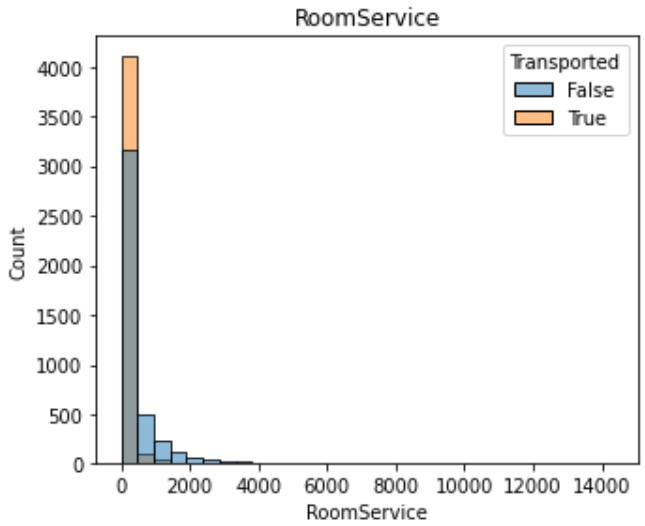
Graphs with the relationship between transported and other cateogories

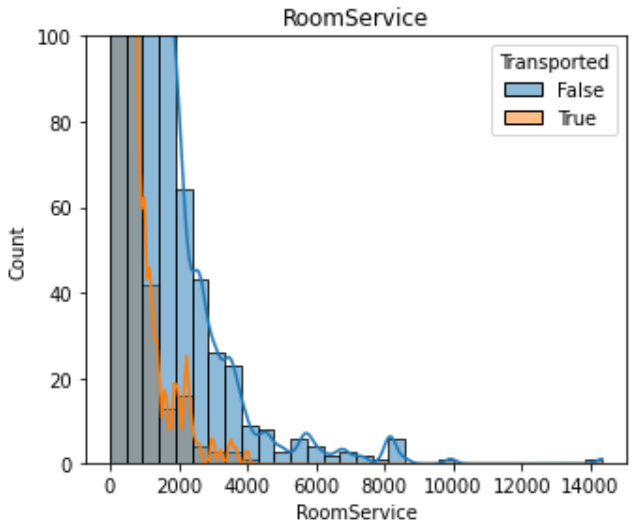
```
## barplot
```

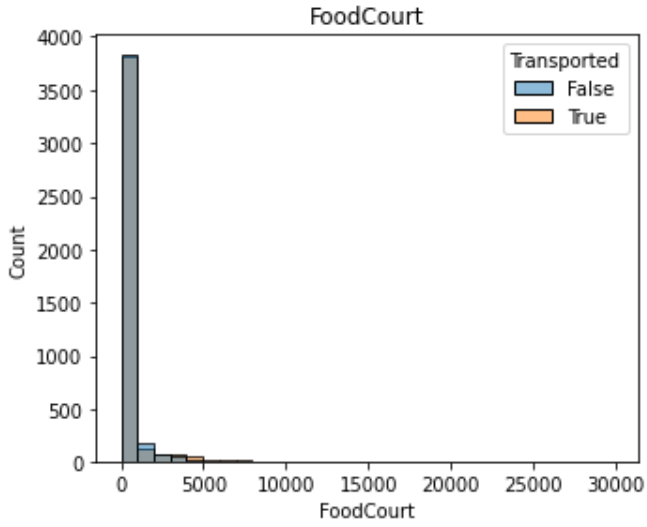
```
plt.figure(figsize=(16,12))

plt.subplot(2,2,1)
totalpl =float(len(ss_train['HomePlanet']))
pl= sns.countplot(data=ss_train, x='HomePlanet', hue='Transported', palette = 'magma')
for p in pl.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totalpl)
    x = p.get_x()
    y = p.get_height()
    pl.annotate(percentage, (x,y))
plt.subplot(2,2,2)
totald = float(len(ss_train['Destination']))
d= sns.countplot(data=ss_train, x='Destination', hue='Transported', palette = 'inferno')
for p in d.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totald)
    x = p.get_x()
    y = p.get_height()
    d.annotate(percentage, (x,y))

plt.subplot(2,2,3)
totalc =float(len(ss_train['CryoSleep']))
c= sns.countplot(data=ss_train, x='CryoSleep', hue='Transported', palette = 'viridis')
for p in c.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totalc)
    x = p.get_x()
    y = p.get_height()
    c.annotate(percentage, (x,y))

plt.subplot(2,2,4)
totalv =float(len(ss_train['VIP']))
v = sns.countplot(data=ss_train, x='VIP', hue='Transported', palette = 'hot')
for p in v.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totalv)
    x = p.get_x()
    y = p.get_height()
    v.annotate(percentage,(x,y))
plt.show()
```

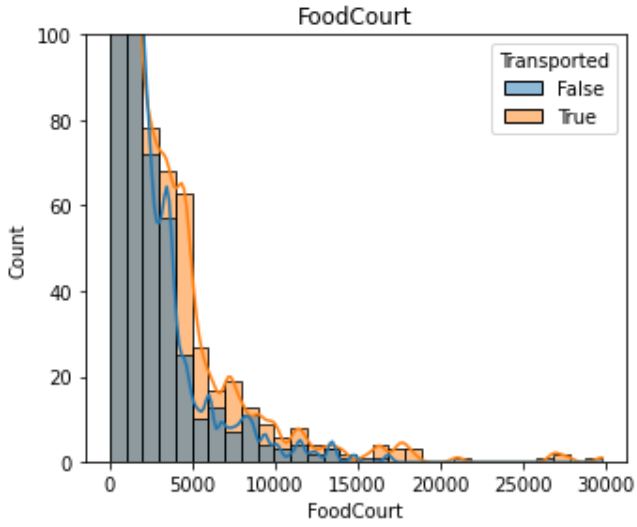


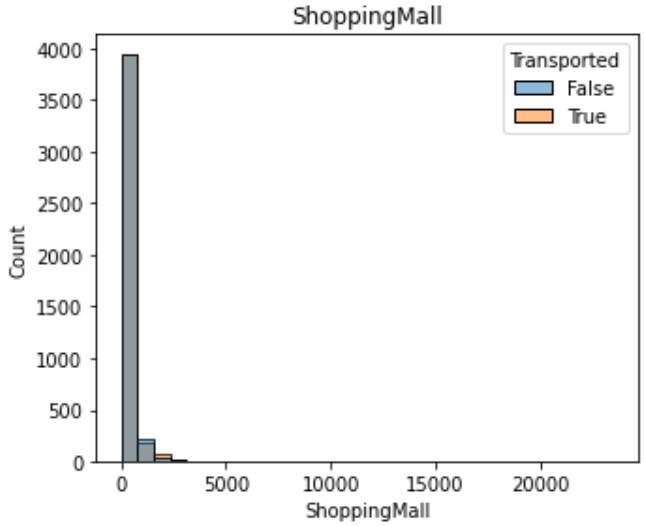

```
numeric_feats=['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']

# Plot expenditure features
fig=plt.figure(figsize=(10,20))
for i, var_name in enumerate(numeric_feats):
    # Left plot
    ax=fig.add_subplot(5,2,2*i+1)
    sns.histplot(data=ss_train, x=var_name, axes=ax, bins=30, kde=False, hue='Transported')
    ax.set_title(var_name)

    # Right plot (truncated)
    ax=fig.add_subplot(5,2,2*i+2)
    sns.histplot(data=ss_train, x=var_name, axes=ax, bins=30, kde=True, hue='Transported')
    plt.ylim([0,100])
    ax.set_title(var_name)
fig.tight_layout()  # Improves appearance a bit
plt.show()
```

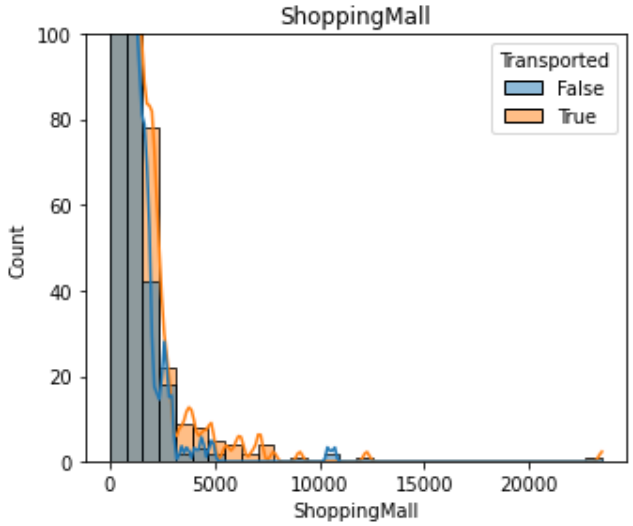

RoomService
Transported
False
True
Count
RoomService

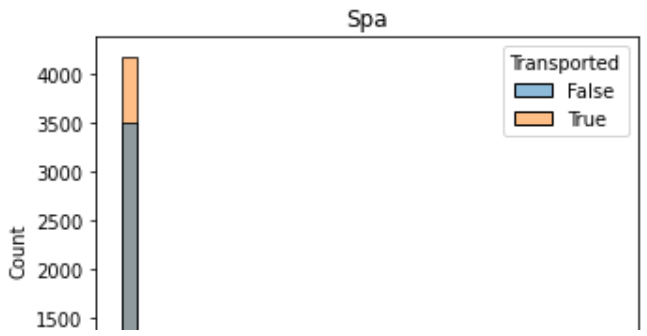

RoomService
Transported
False
True
Count
RoomService

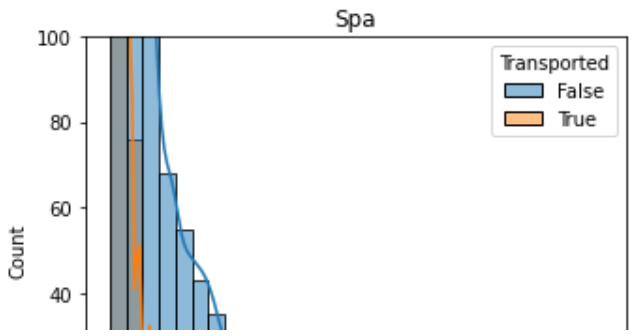

FoodCourt
Transported
False
True
Count
FoodCourt

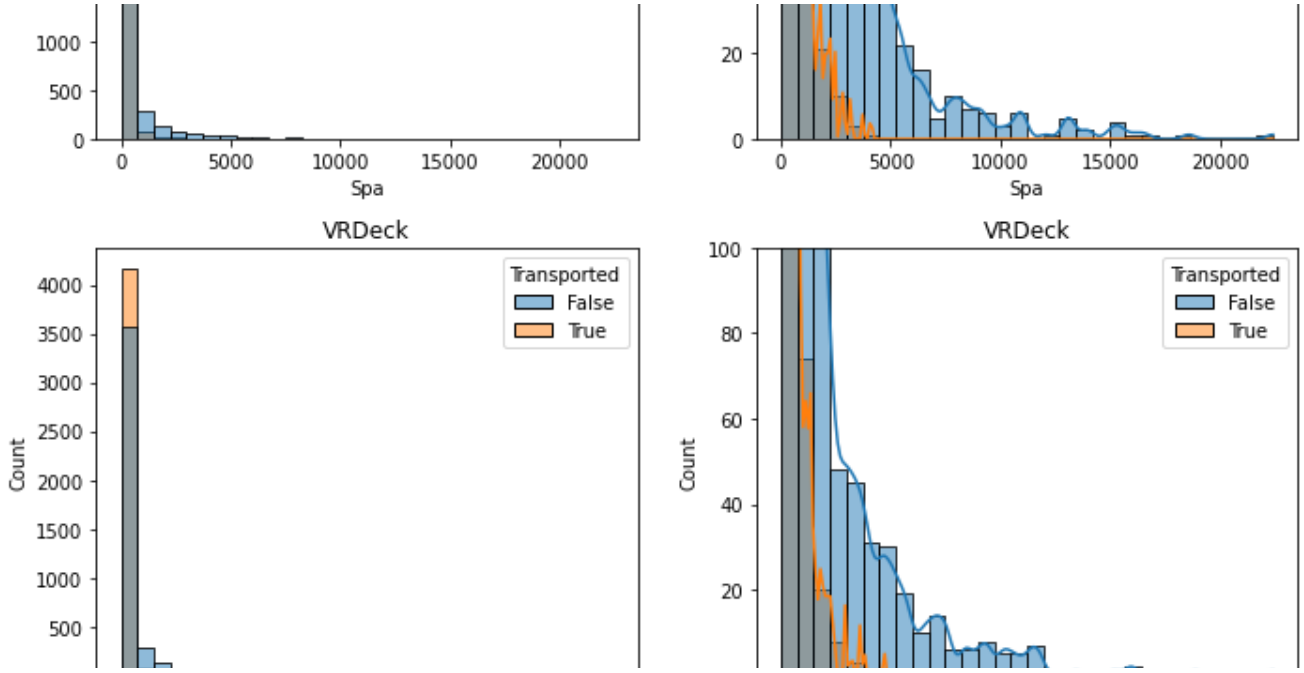

FoodCourt
Transported
False
True
Count
FoodCourt


ShoppingMall
Transported
False
True
Count
ShoppingMall


ShoppingMall
Transported
False
True
Count
ShoppingMall


Spa
Transported
False
True
Count

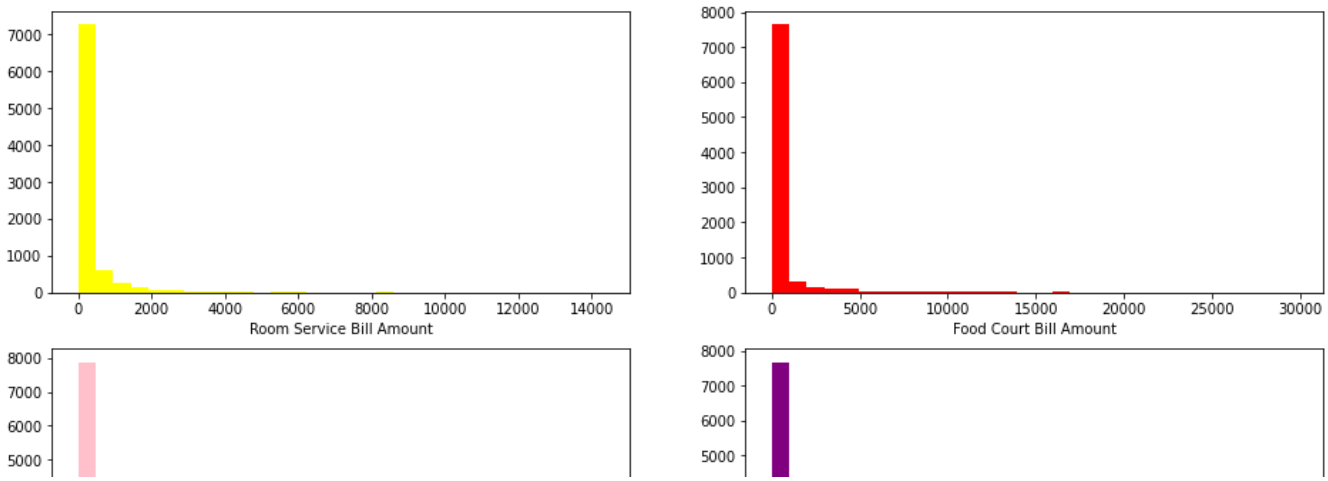

Spa
Transported
False
True
Count



Filling in the Missing Values

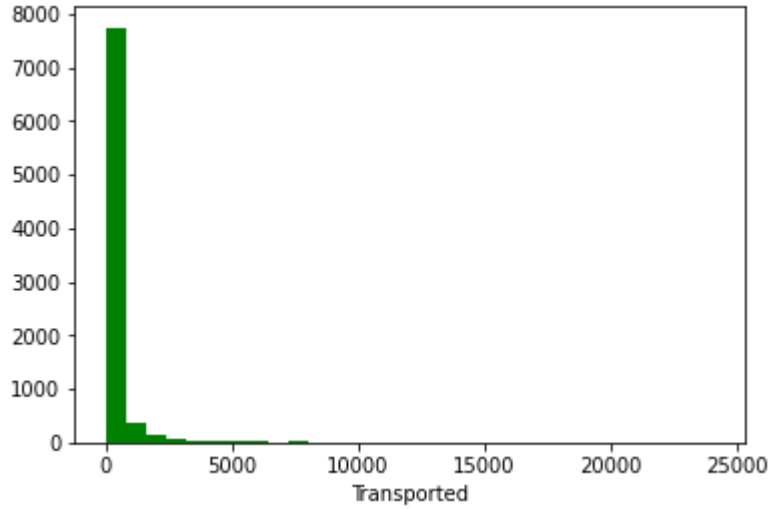
```
## historgram for
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
plt.hist(ss_train.loc[:,'RoomService'], bins = 30, color = 'yellow')

plt.xlabel("Room Service Bill Amount")
plt.subplot(2,2,2)
plt.hist(ss_train.loc[:,'FoodCourt'], bins = 30, color = 'red')
plt.xlabel("Food Court Bill Amount")
plt.subplot(2,2,3)
plt.hist(ss_train.loc[:,'ShoppingMall'], bins = 30, color = 'pink')
plt.xlabel("Shopping Mall Bill Amount")
plt.subplot(2,2,4)
plt.hist(ss_train.loc[:,'Spa'], bins = 30, color = 'purple')
plt.xlabel("Spa Bill Amount")
```

```
Text(0.5, 0, 'Spa Bill Amount')
```




```
plt.hist(ss_train.loc[:,'VRDeck'], bins = 30, color = 'green')
plt.xlabel(xlabel = 'Transported', y = "VR Deck Bill Amount")
```

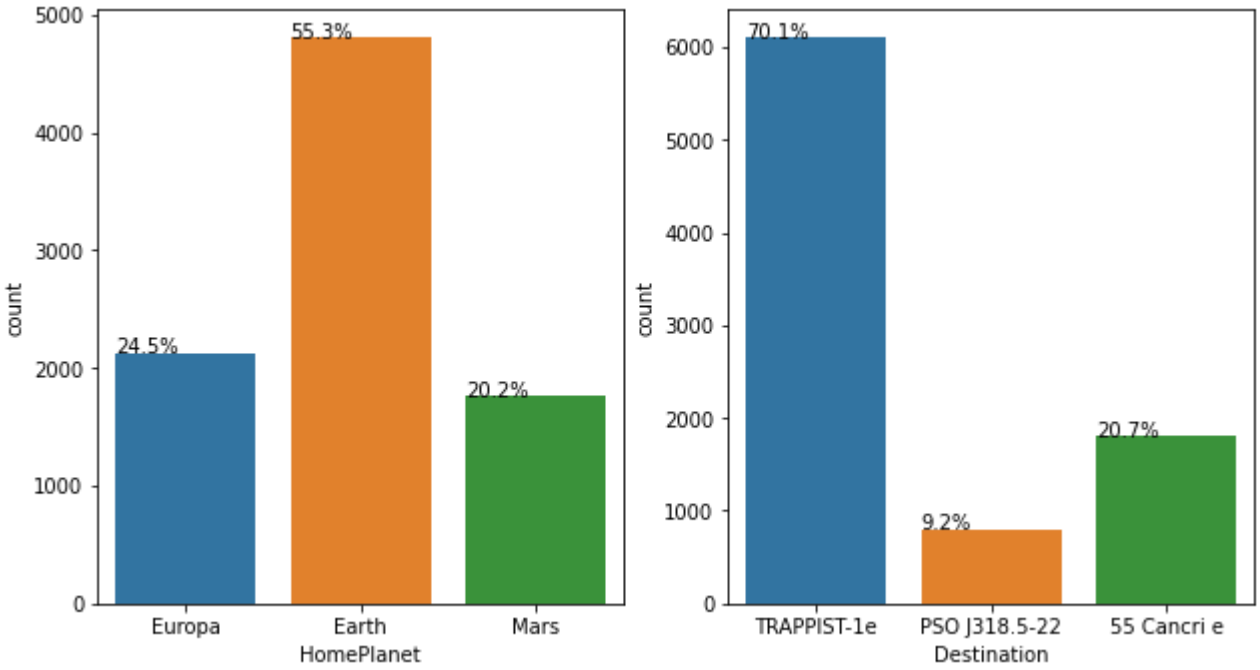
```
Text(0.5, VR Deck Bill Amount, 'Transported')
```



Since age is evenly distributed, which is indicated by the 50 percentile (27) being close to the mean (27), it makes sense to replace missing values in the Age column with the mean value of 27.
For the room service, shopping mall, spa however, the data is not evenly distributed since the summary statistics show that the median value is 0 and the histogram shows the vast majority of values are near 0. Therefore, it makes sense to make the missing values 0 for this column, which happens to be the mode too.

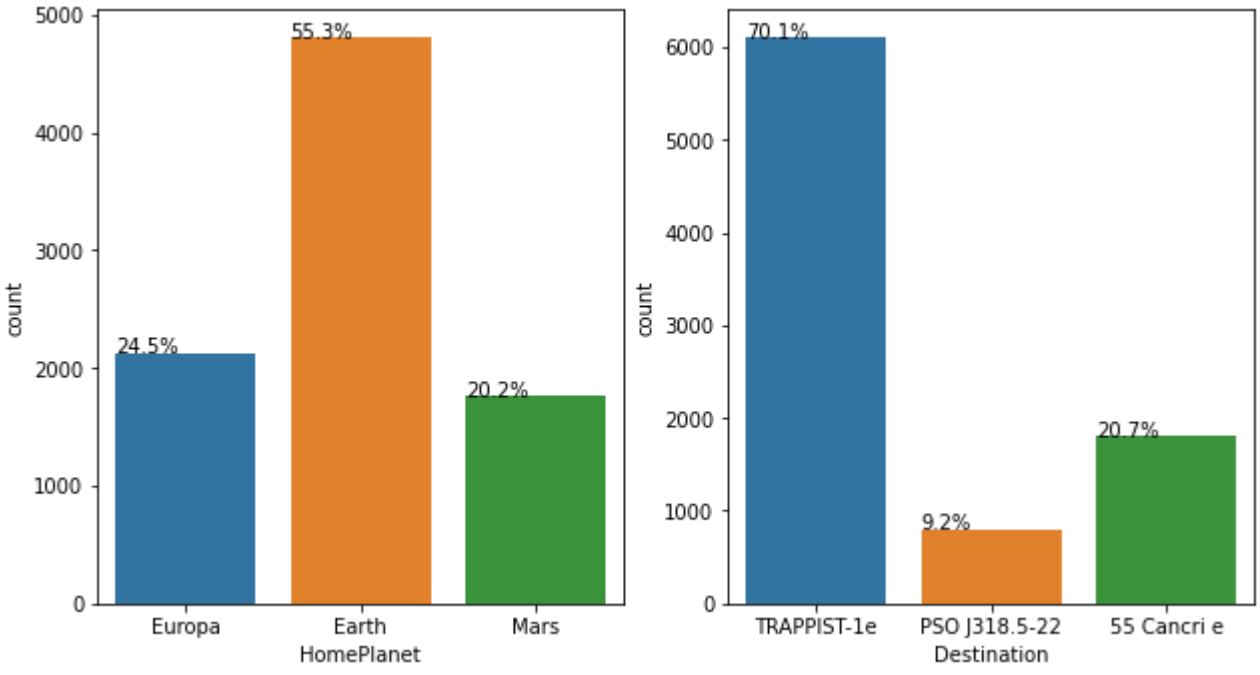
```
for i in ss_train.columns:
    if ss_train[i].isnull().sum().any():
        if i == 'Age':
            ss_train[i] = ss_train[i].fillna(ss_train[i].mean())
            ss_test[i] = ss_test[i].fillna(ss_test[i].mean())
        else:
            ss_train[i] = ss_train[i].fillna(ss_train[i].mode()[0])
            ss_test[i] = ss_test[i].fillna(ss_test[i].mode()[0])
```

```
## create the barplot with the cateroical column
plt.figure(figsize = (16,12))
plt.subplot(2,3,1)
## with checking  difference between with/ without filling missing values.
## with missing value plot
## plot with HomePlanet to see the relation with Homeplanet and Transported
totalh =float(len(ss_train['HomePlanet']))
h = sns.countplot(x = "HomePlanet" , data = ss_train) #fill in missing values of homeplanet w
for p in h.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totalh)
    x = p.get_x()
    y = p.get_height()
    h.annotate(percentage, (x,y))

plt.subplot(2,3,2)
## plot with Destination to see the relation with Destination and Transported
totald =float(len(ss_train['Destination']))
d = sns.countplot(x = "Destination", data = ss_train) #fill in missing values of destination
for p in d.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totald)
    x = p.get_x()
    y = p.get_height()
    d.annotate(percentage,(x,y))

plt.show()
```



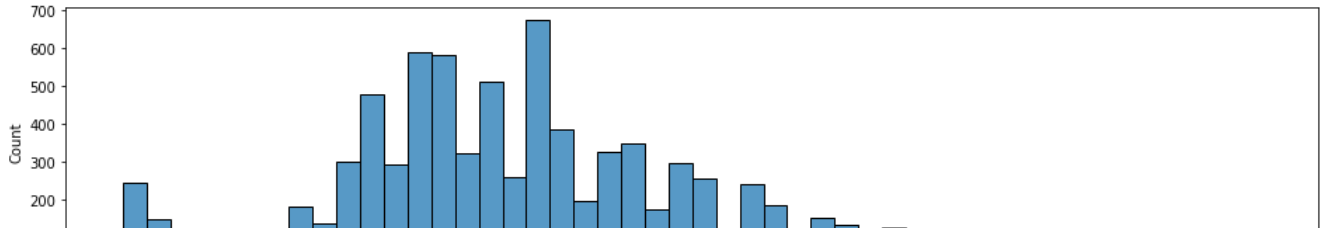

```
ss_train.loc[:,'HomePlanet'] = ss_train.loc[:, 'HomePlanet'].fillna("Earth")
ss_train.loc[:, 'Destination'] = ss_train.loc[:, 'Destination'].fillna("TRAPPIST-1e")
```

```
##for checking the function worked in proper.
##ss_train.isna().sum()
## barplot after we coved with the missing values
plt.figure(figsize = (16,12))
plt.subplot(2,3,1)
totalh =float(len(ss_train['HomePlanet']))
h = sns.countplot(x = "HomePlanet" , data = ss_train) #fill in missing values of homeplanet w
for p in h.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totalh)
    x = p.get_x()
    y = p.get_height()
    h.annotate(percentage, (x,y))

plt.subplot(2,3,2)
totald =float(len(ss_train['Destination']))
d = sns.countplot(x = "Destination", data = ss_train) #fill in missing values of destination
for p in d.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totald)
    x = p.get_x()
    y = p.get_height()
    d.annotate(percentage,(x,y))

plt.show()
```




```
# ## for ages
# figure size
plt.figure(figsize = (16,12))
```

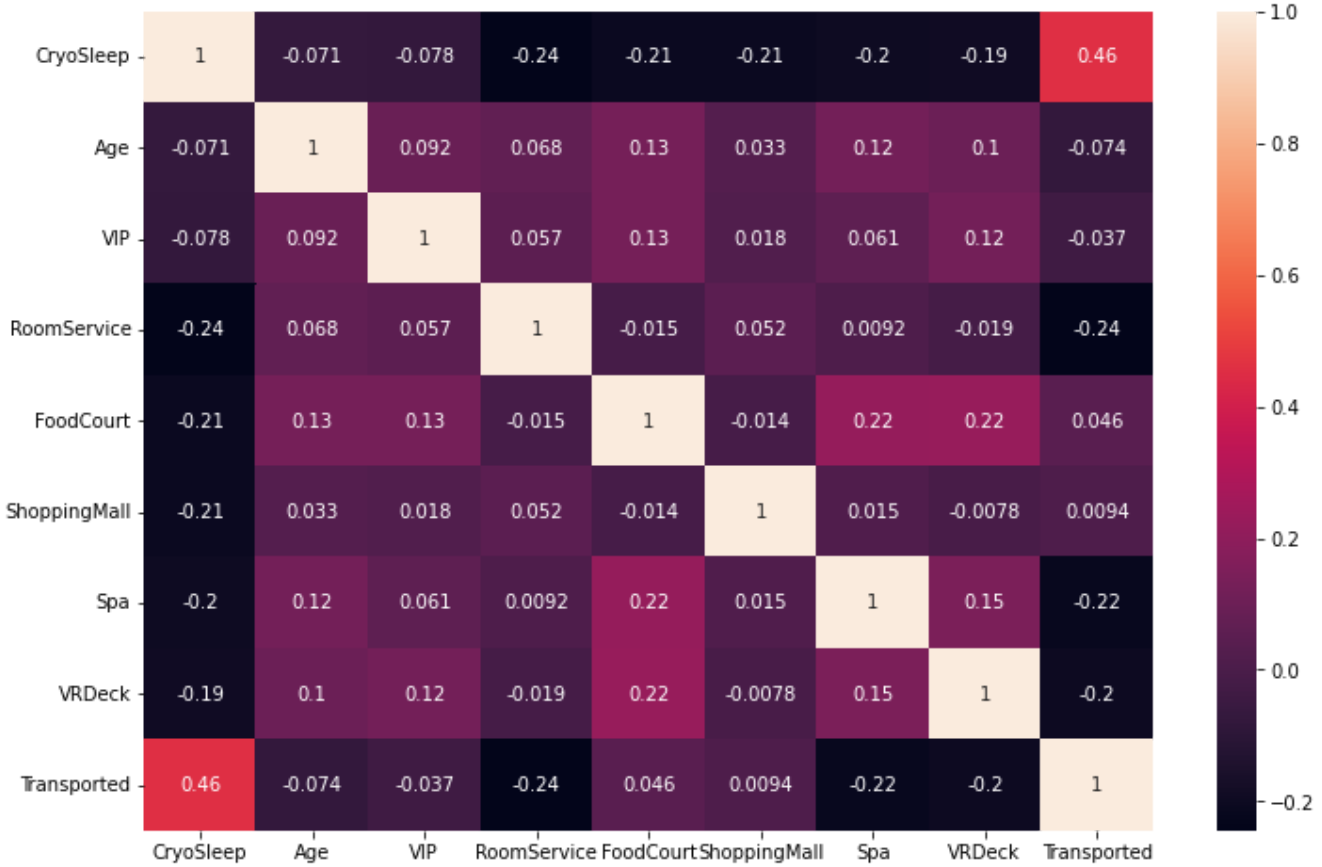
```
## To find the distribution of the Age
plt.subplot(3,1,1)
sns.histplot(data=ss_train, x= 'Age')
plt.subplot(3,1,2)
## to fine the range of the Age
sns.boxplot(data=ss_train, x='Age', palette = ["#fc9272", "#fee0d2"])
plt.subplot(3,1,3)
# For the AgeS reformatt in the 0~10, 11~20 ....
ss_traina = ss_train.copy()
ss_traina['Age_by_decade'] = pd.cut(x=ss_traina['Age'], bins=[9,19,29,39,49,59,69,79,89,99],
# sns.countplot(data = ss_train, x = ss_train['Age_by_decade'], hue='Transported')
## covering all the range of
totala =float(len(ss_traina['Age_by_decade']))
a = sns.countplot(data=ss_traina, x='Age_by_decade', hue='Transported', palette = 'hot')
for p in a.patches:
    percentage ='{:.1f}%'.format(100*p.get_height()/totala)
    x = p.get_x()
    y = p.get_height()
    a.annotate(percentage,(x,y))
plt.show()
```



2.2 Exploratory Data Analysis

Total target distribution

```
# Findning correlations between columns

cor = ss_train.corr()
plt.figure(figsize = (12,8))
sns.heatmap(cor, annot = True)
rel = cor['Transported'].sort_values(ascending = False)
```



The above correlation matrix shows there is not much correlation between the most of the variables. However, there is a slight relationship between room service and transported, spa and

transported, and VRDeck and transported as the relationship is a negative correlation between -.2 and -.25 for these three relationships. We can explore these relationships further with scatter plots.

For PCA, formating the data

```
## feature
## False = 0 True = 1
ss_trainc = ss_train.copy()
ss_testc = ss_test.copy()
```

```
ss_trainc.shape
```

```
(8693, 14)
```

```
ss_train.shape
```

```
(8693, 14)
```

```
ss_train.dtypes
```

```
PassengerId      object
HomePlanet       object
CryoSleep          bool
Cabin            object
Destination      object
Age             float64
VIP                bool
RoomService     float64
FoodCourt       float64
ShoppingMall    float64
Spa             float64
VRDeck          float64
Name             object
Transported        bool
dtype: object
```

```
print(ss_trainc.isna().sum())
```

```
PassengerId     0
HomePlanet      0
CryoSleep       0
Cabin           0
Destination     0
Age             0
VIP             0
RoomService     0
FoodCourt       0
ShoppingMall    0
```

```
Spa             0
VRDeck          0
Name            0
Transported     0
dtype: int64
```

```
## for handling the missing values from the cateorical labels
# Drop qualitative/redundant/high cardinality features
ss_trainc.set_index('PassengerId')
ss_trainc.drop([ 'Cabin', 'Name', 'Age'], axis=1, inplace=True)
```

```
ss_testc.drop(['Cabin', 'Name','Age'], axis=1, inplace=True)
```

```
passengerId = ss_test['PassengerId']
ss_test.set_index('PassengerId')
passengerId
```

```
0       0013_01
1       0018_01
2       0019_01
3       0021_01
4       0023_01
         ...
4272    9266_02
4273    9269_01
4274    9271_01
4275    9273_01
4276    9277_01
Name: PassengerId, Length: 4277, dtype: object
```

```
ss_test.drop(['Cabin', 'Name','Age'], axis=1, inplace=True)
```

```
# Preview resulting training set
ss_trainc.head()
```

	PassengerId	HomePlanet	CryoSleep	Destination	VIP	RoomService	FoodCourt	Shopp
0	0001_01	Europa	False	TRAPPIST-1e	False	0.0	0.0	
1	0002_01	Earth	False	TRAPPIST-1e	False	109.0	9.0	
2	0003_01	Europa	False	TRAPPIST-1e	True	43.0	3576.0	

```
ss_testc.head()
```

	PassengerId	HomePlanet	CryoSleep	Destination	VIP	RoomService	FoodCourt	Shopp
0	0013_01	Earth	True	TRAPPIST-1e	False	0.0	0.0	
1	0018_01	Earth	False	TRAPPIST-1e	False	0.0	9.0	
2	0019_01	Europa	True	55 Cancri e	False	0.0	0.0	

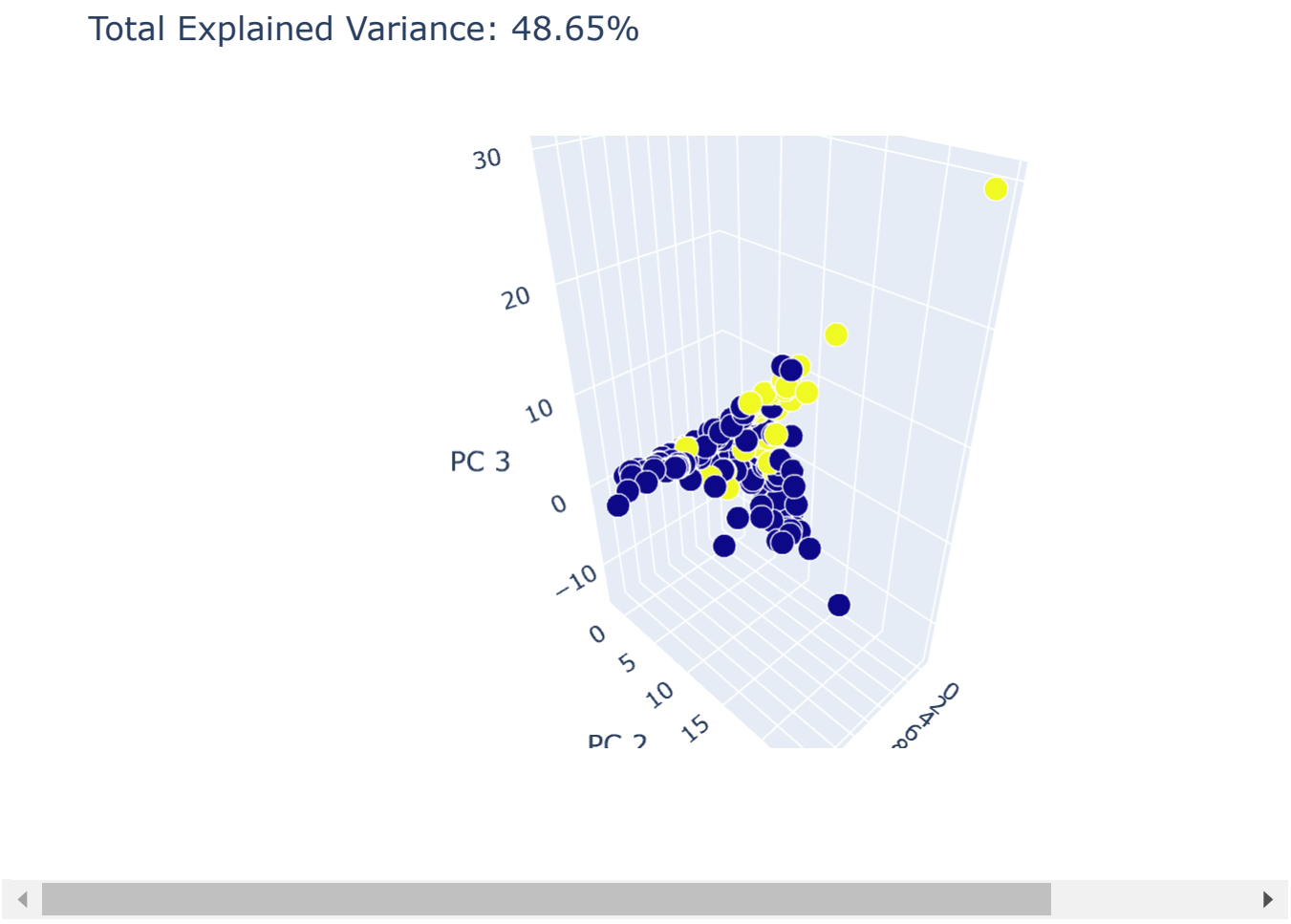
```
y=ss_trainc['Transported'].copy().astype(int)
X=ss_trainc.drop('Transported', axis=1).copy()
X_test=ss_test.copy()


from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.feature_selection import mutual_info_classif
from sklearn.decomposition import PCA
# Indentify numerical and categorical columns
numerical_cols = [cname for cname in X.columns if X[cname].dtype in ['int64', 'float64']]
categorical_cols = [cname for cname in X.columns if X[cname].dtype == "object"]

# Scale numerical data to have mean=0 and variance=1
numerical_transformer = Pipeline(steps=[('scaler', StandardScaler())])

# One-hot encode categorical data
categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(drop='if_binary', handle_u

# Combine preprocessing
ct = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)],
        remainder='passthrough')

# Apply preprocessing
X = ct.fit_transform(X)
X_test = ct.transform(X_test)

# Print new shape
print('Training set shape:', X.shape)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_encoders.py:174: UserWarn
  UserWarning,
Training set shape: (8693, 8706)
```

```
import plotly.express as px
```

```
pca = PCA(n_components=3)
components = pca.fit_transform(X)
print(components.shape)

total_var = pca.explained_variance_ratio_.sum() * 100

fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=y, size=0.1*np.ones(len(X)), opacity = 1,
    title=f'Total Explained Variance: {total_var:.2f}%',
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'},
    width=800, height=500
)
fig.show()
```

```
(8693, 3)
```



The main component explains only 48.65% from original data

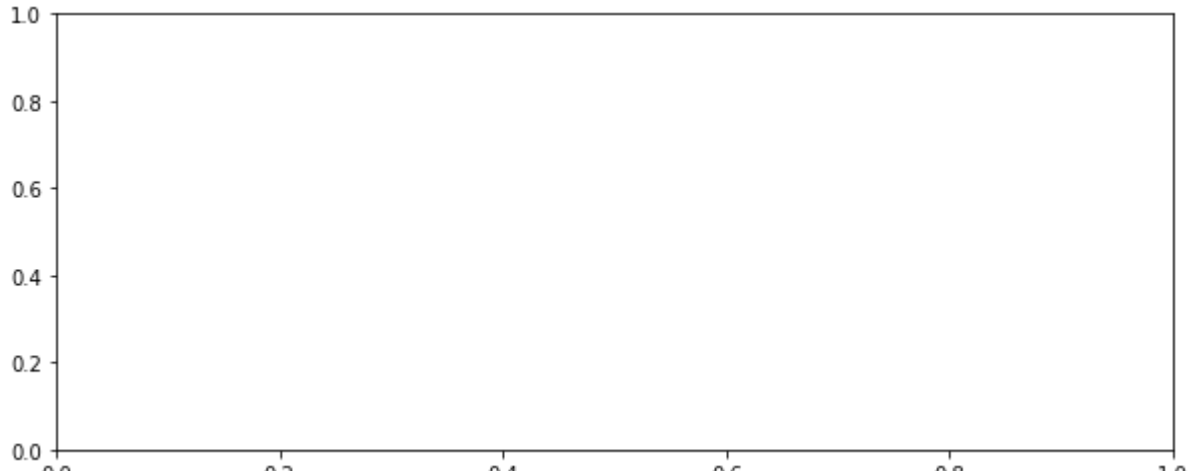
For finding number of main Component

```
# Explained variance (how important each additional principal component is)
#pca = PCA().fit(X)
fig, ax = plt.subplots(figsize=(10,4))
xi = np.arange(1, 1+X.shape[1], step=1)
```

```
yi = np.cumsum(pca.explained_variance_ratio_)
plt.plot(xi, yi, marker='o', linestyle='--', color='b')

# Aesthetics
plt.ylim(0.0,1.1)
plt.xlabel('Number of Components')
plt.xticks(np.arange(1, 1+X.shape[1], step=2))
plt.ylabel('Cumulative variance (%)')
plt.title('Explained variance by each component')
plt.axhline(y=1, color='r', linestyle='-')
plt.text(0.5, 0.85, '100% cut-off threshold', color = 'red')
ax.grid(axis='x')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-33-68f9e50c2e48> in <module>()
      4 xi = np.arange(1, 1+X.shape[1], step=1)
      5 yi = np.cumsum(pca.explained_variance_ratio_)
----> 6 plt.plot(xi, yi, marker='o', linestyle='--', color='b')
      7
      8 # Aesthetics

                                  3 frames
/usr/local/lib/python3.7/dist-packages/matplotlib/axes/_base.py in _plot_args(self,
tup, kwargs)
    340
    341         if x.shape[0] != y.shape[0]:
--> 342             raise ValueError(f"x and y must have same first dimension, but "
    343                              f"have shapes {x.shape} and {y.shape}")
    344         if x.ndim > 2 or y.ndim > 2:

ValueError: x and y must have same first dimension, but have shapes (8706,) and (3,)
```

SEARCH STACK OVERFLOW



Morethan 7 main component describe the most of distrbution.

2.3 Modeling

classifier KNN, NaiveBase, Logistic Regression

I just do it once, if you want we can do it in seperate for the method

To briefly mention the algorithms we will use,

Logistic Regression: Unlike linear regression which uses Least Squares, this model uses Maximum Likelihood Estimation to fit a sigmoid-curve on the target variable distribution. The sigmoid/logistic curve is commonly used when the data is questions had binary output.

K-Nearest Neighbors (KNN): KNN works by selecting the majority class of the k-nearest neighbours, where the metric used is usually Euclidean distance. It is a simple and effective algorithm but can be sensitive by many factors, e.g. the value of k, the preprocessing done to the data and the metric used.

Naive Bayes (NB): Naive Bayes learns how to classify samples by using Bayes' Theorem. It uses prior information to 'update' the probability of an event by incoorporateing this information according to Bayes' law. The algorithm is quite fast but a downside is that it assumes the input features are independent, which is not always the case.

```
## feature (change the string to the numeric)
from sklearn.preprocessing import StandardScaler, LabelEncoder
la = LabelEncoder()
for i in ss_test.columns:
    if ss_train[i].dtype == 'object' or ss_train[i].dtype == 'bool':
        ss_train[i] = la.fit_transform(ss_train[i])
        ss_test[i] = la.fit_transform(ss_test[i])

ss_train['Transported'] = la.fit_transform(ss_train['Transported'])


ss_train.head()
```

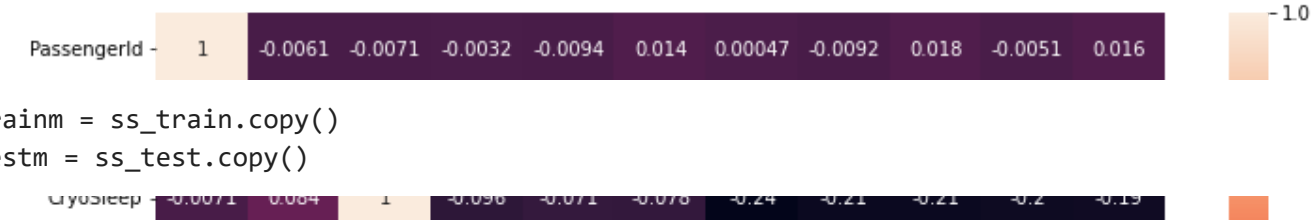
	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodC
0	0	1	0	B/0/P	2	39.0	0	0.0	

```
ss_test.head()
```

	PassengerId	HomePlanet	CryoSleep	Destination	VIP	RoomService	FoodCourt	Shoppi
0	0	0	1	2	0	0.0	0.0	
1	1	0	0	2	0	0.0	9.0	
2	2	1	1	0	0	0.0	0.0	
3	3	1	0	2	0	0.0	6652.0	
4	4	0	0	2	0	10.0	0.0	

```
# Findning correlations between columns
ss_trainco = ss_train.copy()
ss_trainco.drop('Transported', axis = 1, inplace= True)
cor = ss_trainco.corr()
plt.figure(figsize = (12,8))
sns.heatmap(cor, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f77025fc350>
```




```
ss_trainm = ss_train.copy()
ss_testm = ss_test.copy()
```



```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn import tree
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
#from catboost import CatBoostClassifier
import time
```

```
features = ['HomePlanet','CryoSleep','Destination','VIP', 'RoomService', 'FoodCourt', 'Shoppi
y = ss_trainm['Transported']
X = ss_trainm.loc[:, features]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0,shuf



Knn = KNeighborsClassifier()
Logistic_regression = LogisticRegression(solver='lbfgs', max_iter=1000)
Naive_bayes = GaussianNB()
Decision_Tree = tree.DecisionTreeClassifier()
Random_forest = RandomForestClassifier(bootstrap= False, n_estimators=529, max_depth=15, min_
XGB = XGBClassifier(gamma= 0.8151728866167003,learning_rate= 0.031628174313413464,max_depth=
LGBM = LGBMClassifier(learning_rate=0.04183147620569966,max_depth= 25, min_child_samples= 117
GradientBoost = GradientBoostingClassifier(n_estimators=100)
#CatBoost = CatBoostClassifier(objective='CrossEntropy', colsample_bylevel= 0.075879454763026
NNMLP =MLPClassifier(hidden_layer_sizes = (20,20,), activation='relu', solver='adam',max_iter
models = [Knn,Logistic_regression, Naive_bayes , Decision_Tree, Random_forest, XGB, LGBM, Gra
accuracy =[]
train_time =[]
predict_time =[]
total_time =[]
for model in models:
    start = time.time()
    model.fit(X_train, y_train)

    end_train =time.time()
    y_pred = model.predict(X_test)
    end_predict = time.time()
```

```
    acc_model = round(accuracy_score(y_pred, y_test) * 100, 2)
    accuracy.append(acc_model)
    t_time = round(end_train-start,2)
    train_time.append(t_time)
    p_time = round(end_predict-end_train,2)
    predict_time.append(p_time)
    tt_time = round(end_predict-start,2)
    total_time.append(tt_time)


model_name = ['KNN', 'Logistic Regression', 'Naive Bayes', 'Decision Tree', 'Random_forest',
models_table = pd.DataFrame({'Model name': model_name, 'Accuracy percentage': accuracy, 'Trai
models_table
```

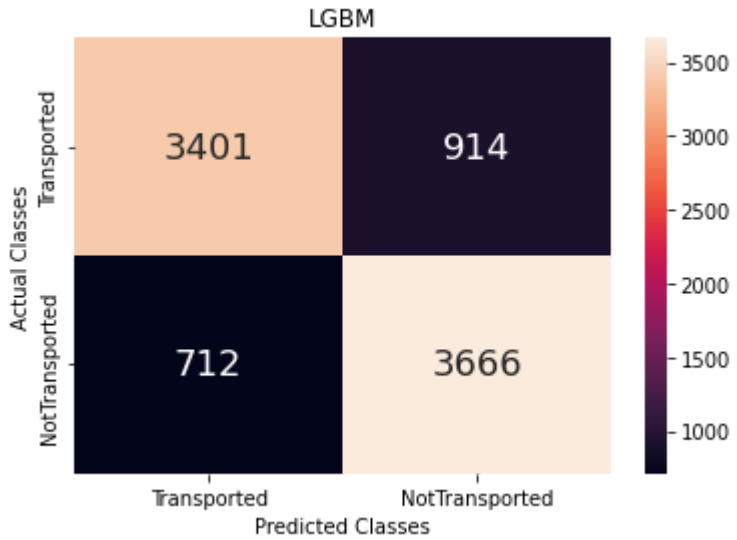
	Model name	Accuracy percentage	Train time	Predict time	Total time
0	KNN	71.71	0.01	0.11	0.12
1	Logistic Regression	78.20	0.10	0.00	0.10
2	Naive Bayes	68.54	0.00	0.00	0.01
3	Decision Tree	74.75	0.02	0.00	0.03
4	Random_forest	78.66	3.72	0.19	3.90
5	XGB	78.29	1.50	0.02	1.52
6	LGBM	78.33	0.35	0.04	0.39
7	GradientBoost	78.33	0.60	0.00	0.61
8	NNMLP	77.32	0.93	0.00	0.94

K-Fold Validation

```
from sklearn.model_selection import cross_val_score ## for K-fold Validation
cv_mean = []
for model in models:
    scores = cross_val_score(model, X, y,
                             cv=5,
                             scoring='accuracy')
    cv_mean.append(scores.mean())

models_validation = pd.DataFrame({'Model name': model_name, 'K-Fold validation mean scores':
models_validation = models_validation.sort_values(by='K-Fold validation mean scores', ascendi
models_validation
```

	Model name	K-Fold validation mean scores
6	LGBM	0.793515
4	Random_forest	0.789258
5	XGB	0.789143
7	GradientBoost	0.788798
1	Logistic Regression	0.786841
8	NNMLP	0.770624
0	KNN	0.749920
3	Decision Tree	0.749454

```
## pick the most accurate one
LR = LGBMClassifier(learning_rate=0.04183147620569966,max_depth= 25, min_child_samples= 117,
LR.fit(X_train, y_train)
LR_prediction = LR.predict(X)

from sklearn.metrics import confusion_matrix  # For find confusion matrix

# Find confusion matrix for this model:
confusion_mat_LR = confusion_matrix(y, LR_prediction)
confusion_mat_dataframe_LR = pd.DataFrame(confusion_mat_LR, index=["Transported", "NotTranspo
sns.heatmap(confusion_mat_dataframe_LR, annot=True, annot_kws={"size": 18}, fmt="d")
plt.title("LGBM")
plt.ylabel('Actual Classes')
plt.xlabel('Predicted Classes')
plt.show()
```




```
from sklearn.metrics import classification_report  # For print evaluation report
report_LR = pd.DataFrame(classification_report(y, LR_prediction, output_dict=True, target_nam
report_LR
```

	Transported	NotTransported	accuracy	macro avg	weighted avg
precision	0.826890	0.800437	0.812953	0.813664	0.813568
recall	0.788181	0.837369	0.812953	0.812775	0.812953
f1-score	0.807072	0.818486	0.812953	0.812779	0.812820
support	4315.000000	4378.000000	0.812953	8693.000000	8693.000000

```
best_prediction = LR_prediction
best_prediction
print(np.round(100*np.round(best_prediction).sum()/len(best_prediction),2), "%")
```

```
52.69 %
```

```
ss_test
```

	PassengerId	HomePlanet	CryoSleep	Destination	VIP	RoomService	FoodCourt	Sho
0	0	0	1	2	0	0.0	0.0	
1	1	0	0	2	0	0.0	9.0	
2	2	1	1	0	0	0.0	0.0	
3	3	1	0	2	0	0.0	6652.0	
4	4	0	0	2	0	10.0	0.0	
...	...	...	...	...	...	...	...	
4272	4272	0	1	2	0	0.0	0.0	
4273	4273	0	0	2	0	0.0	847.0	
4274	4274	2	1	0	0	0.0	0.0	
4275	4275	1	0	2	0	0.0	2680.0	
4276	4276	0	1	1	0	0.0	0.0	

4277 rows × 10 columns

```
print(X_train)
```

```
      HomePlanet  CryoSleep  Destination  VIP  RoomService  FoodCourt  \
3758           2          1            2    0          0.0        0.0
7328           0          0            1    0          0.0        0.0
5550           1          0            0    0          0.0     2427.0
6157           2          0            2    0       1192.0        5.0
1225           0          0            2    0          0.0        0.0
...          ...        ...          ...  ...          ...        ...
4011           0          0            2    0          0.0      252.0
```

```
      1795           0          0           2    0          8.0      652.0
      4668           0          0           2    0          0.0        4.0
      6142           0          0           2    0         89.0        0.0
      3235           0          0           2    0          1.0      467.0

            ShoppingMall     Spa  VRDeck
      3758           0.0     0.0     0.0
      7328         501.0   242.0     0.0
      5550          15.0     5.0     0.0
      6157        1864.0  4287.0     0.0
      1225           0.0     0.0     0.0
      ...            ...     ...     ...
      4011           1.0     0.0   553.0
      1795           0.0     5.0    90.0
      4668         834.0     0.0    32.0
      6142         795.0     0.0     3.0
      3235           4.0     0.0   341.0

      [6519 rows x 9 columns]

  #passengerId = ss_test['PassengerId']
  ss_test = ss_test.drop('PassengerId', axis = 1)
  print(ss_test)

            HomePlanet  CryoSleep  Destination  VIP  RoomService  FoodCourt  \
      0              0          1            2    0          0.0        0.0
      1              0          0            2    0          0.0        9.0
      2              1          1            0    0          0.0        0.0
      3              1          0            2    0          0.0     6652.0
      4              0          0            2    0         10.0        0.0
      ...          ...        ...          ...  ...          ...        ...
      4272           0          1            2    0          0.0        0.0
      4273           0          0            2    0          0.0      847.0
      4274           2          1            0    0          0.0        0.0
      4275           1          0            2    0          0.0     2680.0
      4276           0          1            1    0          0.0        0.0

            ShoppingMall     Spa  VRDeck
      0              0.0     0.0     0.0
      1              0.0  2823.0     0.0
      2              0.0     0.0     0.0
      3              0.0   181.0   585.0
      4            635.0     0.0     0.0
      ...            ...     ...     ...
      4272           0.0     0.0     0.0
      4273          17.0    10.0   144.0
      4274           0.0     0.0     0.0
      4275           0.0     0.0   523.0
      4276           0.0     0.0     0.0

      [4277 rows x 9 columns]



  LR = LGBMClassifier(learning_rate=0.04183147620569966,max_depth= 25, min_child_samples= 117,
```

```
Most = LR.fit(X_train, y_train)
Most.predict(ss_test)
pred1 = pd.DataFrame(Most.predict(ss_test))
pred1.columns =['Transported']
pred1['PassengerId'] = passengerId
pred1 = pred1[['PassengerId', 'Transported']]
for i in range(len(pred1)):
  if pred1.iloc[i,1] == 1:
    pred1.iloc[i,1] = 'True'
  else:
    pred1.iloc[i,1] = 'False'
  #pred1.iloc[i,1] = pred1.iloc[i,1]
pred1.to_csv("ss_submission.csv", index =False)
pred1
```

	PassengerId	Transported
0	0013_01	True
1	0018_01	False
2	0019_01	True
3	0021_01	True
4	0023_01	False
...	...	...
4272	9266_02	True
4273	9269_01	False
4274	9271_01	True
4275	9273_01	True
4276	9277_01	True

4277 rows × 2 columns

Housing Prices Section

2.1 Handing missing values

```
hp_train = pd.read_csv('/content/hp_train.csv') ## this might be on the directory
hp_test = pd.read_csv('/content/hp_test.csv')
hp_train
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCo
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	

1460 rows × 81 columns

```
hp_train.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2

8 rows × 38 columns

```
from sklearn import preprocessing
```

```
categorical = (hp_train.dtypes == object)
categorical1 = (hp_test.dtypes == object)
categorical

for col in range(len(hp_train.columns)):
  if categorical[col] == True :
    le = preprocessing.LabelEncoder()
    hp_train.iloc[:, col] = le.fit_transform(hp_train.iloc[:,col])

for col in range(len(hp_test.columns)):
  if categorical1[col] == True :
    le = preprocessing.LabelEncoder()
    hp_test.iloc[:, col] = le.fit_transform(hp_test.iloc[:,col])
hp_test
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCo
0	1461	20	2	80.0	11622	1	2	3	
1	1462	20	3	81.0	14267	1	2	0	
2	1463	60	3	74.0	13830	1	2	0	
3	1464	60	3	78.0	9978	1	2	0	
4	1465	120	3	43.0	5005	1	2	0	
...	...	...	...	...	...	...	...	...	
1454	2915	160	4	21.0	1936	1	2	3	
1455	2916	160	4	21.0	1894	1	2	3	
1456	2917	20	3	160.0	20000	1	2	3	
1457	2918	85	3	62.0	10441	1	2	3	
1458	2919	60	3	74.0	9627	1	2	3	

1459 rows × 80 columns

2.2 Exploratory Data Analysis

```
corr_matrix = hp_train.corr()
corr_df = pd.DataFrame(corr_matrix["SalePrice"].sort_values(ascending=False))
corr_df
#print(len(corr_matrix))
```

	SalePrice
SalePrice	1.000000
OverallQual	0.790982
GrLivArea	0.708624
GarageCars	0.640409
GarageArea	0.623431
...	...
FireplaceQu	-0.459605
GarageFinish	-0.549247
KitchenQual	-0.589189
BsmtQual	-0.620886
ExterQual	-0.636884

81 rows × 1 columns

```
## correlation matrix  might need correaltion with the sale prices
hp_train_subset = hp_train[["SalePrice","OverallQual","GrLivArea", "GarageCars","GarageArea",
hp_train_subset
hp_test_subset = hp_test[["OverallQual","GrLivArea", "GarageCars","GarageArea","TotalBsmtSF",
hp_test_subset
```

	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath
0	5	896	1.0	730.0	882.0	896	1

```
## finding the missing value
print(hp_train_subset.isna().sum())
##  compare with the missing values from the original data
print(hp_train['SalePrice'].isna().sum())
print(hp_train['OverallQual'].isna().sum())
print(hp_train['GrLivArea'].isna().sum())
print(hp_train['GarageCars'].isna().sum())
print(hp_train['GarageArea'].isna().sum())
print(hp_train['TotalBsmtSF'].isna().sum())
print(hp_train['1stFlrSF'].isna().sum())
print(hp_train['FullBath'].isna().sum())
print(hp_train['TotRmsAbvGrd'].isna().sum())
print(hp_train['YearBuilt'].isna().sum())
print(hp_train['YearRemodAdd'].isna().sum())
print(hp_train['GarageYrBlt'].isna().sum())##
print(hp_train['MasVnrArea'].isna().sum())##
print(hp_train['Fireplaces'].isna().sum())
print(hp_test_subset.isna().sum())
```

```
SalePrice        0
OverallQual      0
GrLivArea        0
GarageCars       0
GarageArea       0
TotalBsmtSF      0
1stFlrSF         0
FullBath         0
TotRmsAbvGrd     0
YearBuilt        0
YearRemodAdd     0
GarageYrBlt     81
MasVnrArea       8
Fireplaces       0
dtype: int64
0
0
0
0
0
0
0
0
0
0
81
8
0
OverallQual      0
GrLivArea        0
```

```
GarageCars       1
GarageArea       1
TotalBsmtSF      1
1stFlrSF         0
FullBath         0
TotRmsAbvGrd     0
YearBuilt        0
YearRemodAdd     0
GarageYrBlt     78
MasVnrArea      15
Fireplaces       0
dtype: int64
```

```
hp_test_subset.isna().sum()
#len(hp_test_subset)
hp_test_subset['GarageCars'] = hp_test_subset['GarageCars'].replace(np.nan, np.nanmean(hp_tes
hp_test_subset['GarageArea'] = hp_test_subset['GarageArea'].replace(np.nan, np.nanmean(hp_tes
hp_test_subset['TotalBsmtSF'] = hp_test_subset['TotalBsmtSF'].replace(np.nan, np.nanmean(hp_t

#hp_test_subset = hp_test_subset[hp_test_subset['TotalBsmtSF'].notna()]

hp_test_subset['GarageYrBlt'] = hp_test_subset['GarageYrBlt'].replace(np.nan, np.nanmean(hp_t
hp_test_subset['MasVnrArea'] = hp_test_subset['MasVnrArea'].replace(np.nan,0)
hp_test_subset.isna().sum()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  """
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  if __name__ == '__main__':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  # Remove the CWD from sys.path while we load stuff.
OverallQual     0
GrLivArea       0
GarageCars      0
GarageArea      0
TotalBsmtSF     0
1stFlrSF        0
FullBath        0
TotRmsAbvGrd    0
YearBuilt       0
YearRemodAdd    0
GarageYrBlt     0
MasVnrArea      0
Fireplaces      0
dtype: int64
```

```
import statistics
hp_train_subset.describe()
```

	SalePrice	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	14
mean	180921.195890	6.099315	1515.463699	1.767123	472.980137	1057.429452	1
std	79442.502883	1.382997	525.480383	0.747315	213.804841	438.705324	3
min	34900.000000	1.000000	334.000000	0.000000	0.000000	0.000000	3
25%	129975.000000	5.000000	1129.500000	1.000000	334.500000	795.750000	8
50%	163000.000000	6.000000	1464.000000	2.000000	480.000000	991.500000	1(
75%	214000.000000	7.000000	1776.750000	2.000000	576.000000	1298.250000	1:
max	755000.000000	10.000000	5642.000000	4.000000	1418.000000	6110.000000	4(

```
## missing value check
print('TRAIN SET MISSING VALUES:')
print(hp_train.isna().sum())
print('')
print('TEST SET MISSING VALUES:')
print(hp_test.isna().sum())
```

```
TRAIN SET MISSING VALUES:
Id                0
MSSubClass        0
MSZoning          0
```

```
LotFrontage      259
LotArea            0
                 ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64

TEST SET MISSING VALUES:
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      227
LotArea            0
                 ...
MiscVal            0
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
Length: 80, dtype: int64
```

```
plt.figure(figsize=(16,8))

plt.subplot(2,2,1)
plt.title("Gross Living Area and Sale Price")
line_params = np.polyfit(hp_train.loc[:,'GrLivArea'], hp_train.loc[:,'SalePrice'], 1)
line = line_params[1] + line_params[0] * hp_train.loc[:,'GrLivArea']
plt.plot(hp_train.loc[:,'GrLivArea'], line, 'r')
plt.scatter(x = hp_train.loc[:,'GrLivArea'],y = hp_train.loc[:,'SalePrice'] , color = 'yellow

plt.subplot(2,2,2)
plt.title("Year Built and Sale Price")
line_params = np.polyfit(hp_train.loc[:,'YearBuilt'], hp_train.loc[:,'SalePrice'], 1)
line = line_params[1] + line_params[0] * hp_train.loc[:,'YearBuilt']
plt.plot(hp_train.loc[:,'YearBuilt'], line, 'r')
plt.scatter(x = hp_train.loc[:,'YearBuilt'],y= hp_train.loc[:,'SalePrice'] , color = 'orange'

plt.subplot(2,2,3)
plt.title("Garage Area and Sale Price")
line_params = np.polyfit(hp_train.loc[:,'GarageArea'], hp_train.loc[:,'SalePrice'], 1)
line = line_params[1] + line_params[0] * hp_train.loc[:,'GarageArea']
plt.plot(hp_train.loc[:,'GarageArea'], line, 'r')
plt.scatter(x = hp_train.loc[:,'GarageArea'],y= hp_train.loc[:,'SalePrice'] , color = 'purple

plt.subplot(2,2,4)
plt.title("Year of Remodel and Sale Price")
line_params = np.polyfit(hp_train.loc[:,'YearRemodAdd'], hp_train.loc[:,'SalePrice'], 1)
```

```
line = line_params[1] + line_params[0] * hp_train.loc[:,'YearRemodAdd']
plt.plot(hp_train.loc[:,'YearRemodAdd'], line, 'r')
plt.scatter(x = hp_train.loc[:,'YearRemodAdd'],y = hp_train.loc[:,'SalePrice'] , color = 'gre
```

```
<matplotlib.collections.PathCollection at 0x7f76f7d82210>
```

Understanding Attribute Distributions

```
print(hp_train_subset.head(5))
plt.figure(figsize = (16,12))

plt.subplot(2,4,1)
sns.distplot(hp_train.loc[:,'SalePrice'], color = 'red')

plt.subplot(2,4,2)
sns.distplot(hp_train.loc[:,'GrLivArea'], color = 'blue')

plt.subplot(2,4,3)
sns.distplot(hp_train.loc[:,'YearBuilt'], color = 'yellow')

plt.subplot(2,4,4)
sns.distplot(hp_train.loc[:,'YearRemodAdd'], color = 'green')
```

```
plt.subplot(2,4,5)
plt.hist(hp_train.loc[:,'OverallQual'], color = 'orange')

plt.subplot(2,4,6)
plt.hist(hp_train.loc[:,'GarageCars'], color = 'blue')

plt.subplot(2,4,7)
plt.hist(hp_train.loc[:,'FullBath'], color = 'pink')

plt.subplot(2,4,8)
plt.hist(hp_train.loc[:,'TotRmsAbvGrd'], color = 'purple')
```

```
   SalePrice  OverallQual  GrLivArea  GarageCars  GarageArea  TotalBsmtSF  \
0     208500            7       1710           2         548          856
1     181500            6       1262           2         460         1262
2     223500            7       1786           2         608          920
3     140000            7       1717           3         642          756
4     250000            8       2198           3         836         1145

   1stFlrSF  FullBath  TotRmsAbvGrd  YearBuilt  YearRemodAdd  GarageYrBlt  \
0       856         2             8       2003          2003       2003.0
1      1262         2             6       1976          1976       1976.0
2       920         2             6       2001          2002       2001.0
3       961         1             7       1915          1970       1998.0
4      1145         2             9       2000          2000       2000.0

   MasVnrArea  Fireplaces
0       196.0           0
1         0.0           1
2       162.0           1
3         0.0           1
4       350.0           1
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
(array([ 18.,  97., 275., 402., 329., 262.,  47.,  18.,  11.,   1.]),
 array([ 2. ,  3.2,  4.4,  5.6,  6.8,  8. ,  9.2, 10.4, 11.6, 12.8, 14. ]),
 <a list of 10 Patch objects>)
```

```
import statistics
hp_train_subset['GarageYrBlt'] = hp_train_subset['GarageYrBlt'].replace(np.nan, np.nanmean(hp
hp_train_subset['MasVnrArea'] = hp_train_subset['MasVnrArea'].replace(np.nan,0)
hp_train_subset.isna().sum()
```

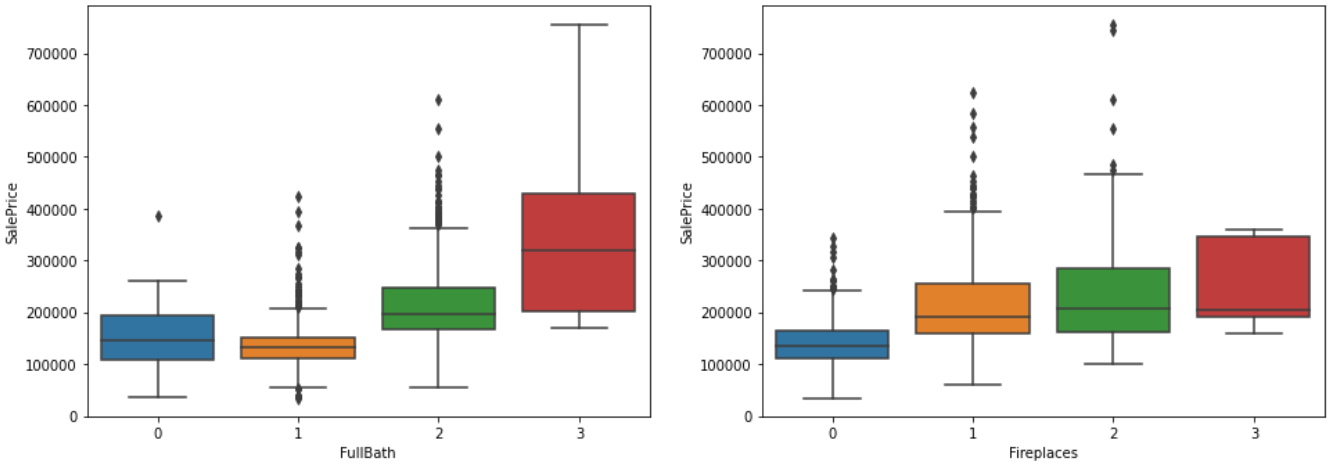
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  This is separate from the ipykernel package so we can avoid doing imports until
SalePrice       0
OverallQual     0
GrLivArea       0
GarageCars      0
GarageArea      0
TotalBsmtSF     0
1stFlrSF        0
```

```
FullBath        0
TotRmsAbvGrd    0
YearBuilt       0
YearRemodAdd    0
GarageYrBlt     0
MasVnrArea      0
Fireplaces      0
dtype: int64
```

Sale price has a slightly right skewed normal distribution

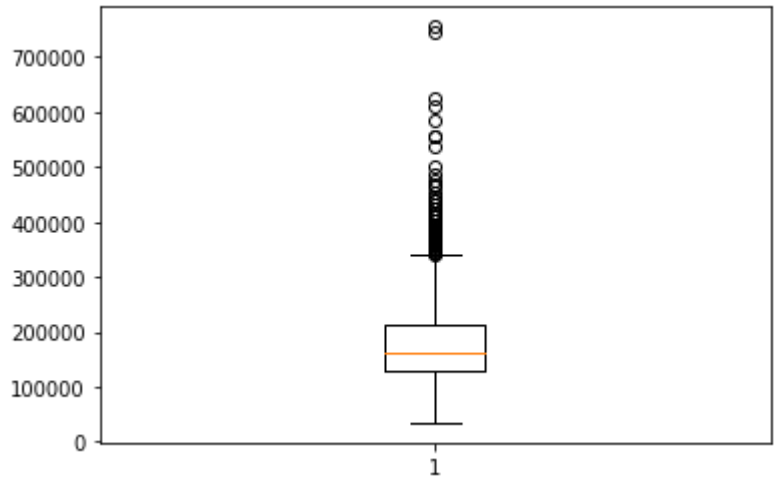
```
plt.figure(figsize=(16,12))

plt.subplot(2,2,1)
sns.boxplot(x = hp_train["FullBath"], y = hp_train["SalePrice"])

plt.subplot(2,2,2)
sns.boxplot(x = hp_train["Fireplaces"], y = hp_train["SalePrice"])

plt.subplot(2,2,3)
sns.boxplot(x = hp_train["TotRmsAbvGrd"], y = hp_train["SalePrice"])

plt.subplot(2,2,4)
sns.boxplot(x = hp_train["OverallQual"], y = hp_train["SalePrice"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f76f789d4d0>
```




```
plt.boxplot(hp_train.loc[:,'SalePrice'])
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x7f76f753c250>],
 'caps': [<matplotlib.lines.Line2D at 0x7f76f7541210>,
  <matplotlib.lines.Line2D at 0x7f76f7541750>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f76f7547250>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7f76f7541cd0>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7f76f753c750>,
  <matplotlib.lines.Line2D at 0x7f76f753cc90>]}
```



```
#standardize the columns
import statistics
for col in range(len(hp_train_subset.columns) - 1):
  mean = hp_train_subset.iloc[:,col].mean()
  std = statistics.stdev(hp_train_subset.iloc[:,col])
  for row in range(len(hp_train_subset.columns)):
    hp_train_subset.iloc[row,col] = hp_train_subset.iloc[row,col] - mean / std
for col in range(len(hp_test_subset.columns) - 1):
  mean = hp_test_subset.iloc[:,col].mean()
  std = statistics.stdev(hp_test_subset.iloc[:,col])
  for row in range(len(hp_test_subset.columns)):
    hp_test_subset.iloc[row,col] = hp_test_subset.iloc[row,col] - mean / std
hp_test_subset
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  self._setitem_single_column(loc, value, pi)
```

	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBat
0	0.769229	892.93956	0.887345	727.821082	879.637209	893.095344	-1.82955
1	1.769229	1325.93956	0.887345	309.821082	1326.637209	1326.095344	-1.82955
2	0.769229	1625.93956	1.887345	479.821082	925.637209	925.095344	-0.82955
3	1.769229	1600.93956	1.887345	467.821082	923.637209	923.095344	-0.82955
4	3.769229	1276.93956	1.887345	503.821082	1277.637209	1277.095344	-0.82955
...	...	...	...	...	...	...	
1454	4.000000	1092.00000	0.000000	0.000000	546.000000	546.000000	1.00000
1455	4.000000	1092.00000	1.000000	286.000000	546.000000	546.000000	1.00000
1456	5.000000	1224.00000	2.000000	576.000000	1224.000000	1224.000000	1.00000
1457	5.000000	970.00000	0.000000	0.000000	912.000000	970.000000	1.00000
1458	7.000000	2000.00000	3.000000	650.000000	996.000000	996.000000	2.00000

1459 rows × 13 columns

2.3 Modeling

Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score


hp_train_subset = hp_train_subset.reindex(columns = ["OverallQual","GrLivArea", "GarageCars",

hp_X_train = hp_train_subset.iloc[:,0:len(hp_train_subset.columns) - 1]
hp_y_train = hp_train_subset.iloc[:,len(hp_train_subset.columns) - 1]
reg = LinearRegression()
```

```
lr_scores = cross_val_score(reg, hp_X_train, hp_y_train, cv=5)

hp_train = pd.read_csv('/content/hp_train.csv')
hp_X_train_full = hp_train.iloc[:, 0:len(hp_train.columns) - 1]
hp_y_train_full = hp_train.iloc[:, len(hp_train.columns) - 1]

reg = LinearRegression().fit(hp_X_train, hp_y_train)
pred = pd.DataFrame(reg.predict(hp_test_subset))
pred.columns = ['SalePrice']
pred['Id'] = hp_test['Id']
hp_train_subset.isna().sum()
pred = pred.reindex(columns = ['Id','SalePrice'])
print(pred)
pred.to_csv("submission.csv", index=False)
#print(scores)
```

```
          Id      SalePrice
0     1461   58114.927867
1     1462   89835.112915
2     1463  103810.539143
3     1464  117212.228092
4     1465  132953.982061
...    ...            ...
1454  2915   81046.712429
1455  2916   93208.511338
1456  2917  161259.328396
1457  2918  111863.312787
1458  2919  232832.633124

[1459 rows x 2 columns]
```

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

rfg = RandomForestRegressor().fit(hp_X_train,hp_y_train )
rfg_scores = cross_val_score(rfg, hp_X_train, hp_y_train, cv=5)
pred1 = pd.DataFrame(rfg.predict(hp_test_subset))
pred1.columns = ['SalePrice']
pred1['Id'] = hp_test['Id']
pred1 = pred1[['Id','SalePrice']]
pred1.to_csv("submission.csv", index=False)

pred1
rfg_scores
```

```
array([0.76807923, 0.78449284, 0.87399543, 0.86700697, 0.80936675])
```

Support Vector Regressor

```
from sklearn.svm import SVR

svr_scores = cross_val_score(SVR(), hp_X_train, hp_y_train, cv=5)
svr = SVR().fit(hp_X_train, hp_y_train)
svr_pred = pd.DataFrame(svr.predict(hp_test_subset))
print(svr_scores)
svr_pred
```

```
[-0.06927692 -0.05999983 -0.0551195  -0.01513612 -0.05462687]
```

	0
0	162936.387675
1	162999.134207
2	162988.573970
3	162986.255755
4	162995.021121
...	...
1454	162906.776748
1455	162912.508115
1456	162988.323566
1457	162927.521726
1458	163032.714801

1459 rows × 1 columns

Gradient Boosting Regressor

```
from sklearn.ensemble import GradientBoostingRegressor

gbreg = GradientBoostingRegressor().fit(hp_X_train, hp_y_train)
gbr_scores = cross_val_score(GradientBoostingRegressor(), hp_X_train, hp_y_train, cv=5)

pred = pd.DataFrame(gbreg.predict(hp_test_subset))
pred.columns = ['SalePrice']
pred['Id'] = hp_test['Id']
hp_train_subset.isna().sum()
```

```
pred = pred.reindex(columns = ['Id','SalePrice'])
print(pred)
pred.to_csv("submission.csv", index=False)
```

```
             Id      SalePrice
     0     1461  134410.831533
     1     1462  156822.989297
     2     1463  208628.594480
     3     1464  209180.132000
     4     1465  154486.149391
     ...    ...            ...
     1454  2915   77680.706957
     1455  2916   86527.857133
     1456  2917  164220.687821
     1457  2918  125396.013922
     1458  2919  240039.416163

     [1459 rows x 2 columns]
```

Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor

dt_scores = cross_val_score(DecisionTreeRegressor(), hp_X_train, hp_y_train, cv = 5)
```

Bayesian Ridge Regression

```
from sklearn.linear_model import BayesianRidge

br_scores = cross_val_score(BayesianRidge(), hp_X_train, hp_y_train,cv = 5)
np.mean(br_scores)
```

```
     0.7218043323668085
```

Lasso Regression

```
from sklearn import linear_model


lass_scores = cross_val_score(linear_model.Lasso(alpha = .1), hp_X_train, hp_y_train,cv = 5)

lass_scores
```

```
array([0.63444421, 0.77987323, 0.78212899, 0.79386776, 0.60062576])
```

MLP Regressor

```
from sklearn.neural_network import MLPRegressor

mlp_scores = cross_val_score(MLPRegressor(max_iter = 1000), hp_X_train, hp_y_train, cv = 5)
mlp_scores
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py
  ConvergenceWarning,
array([0.75319014, 0.7157468 , 0.72456153, 0.6903468 , 0.53314154])
```

KNN Regressor

```
from sklearn import neighbors
knn_scores = cross_val_score(neighbors.KNeighborsRegressor(n_neighbors = 5) , hp_X_train, hp_

knn_scores
```

```
array([0.72401033, 0.70709032, 0.78441373, 0.73140826, 0.6622646 ])
```

```
df_scores = pd.DataFrame(columns = ['Model','Score'])
df_scores['Model'] = ['Linear Regression', 'Random Forest Regression','Support Vector Regress
df_scores['Score'] = [np.mean(lr_scores), np.mean(rfg_scores), np.mean(svr_scores), np.mean(g
#df_scores.loc[0,'Model'] = 'Linear Regression'
#df_scores.loc[1,'Model']
df_scores #.sort_values(by = 'Score', ascending = False)
```