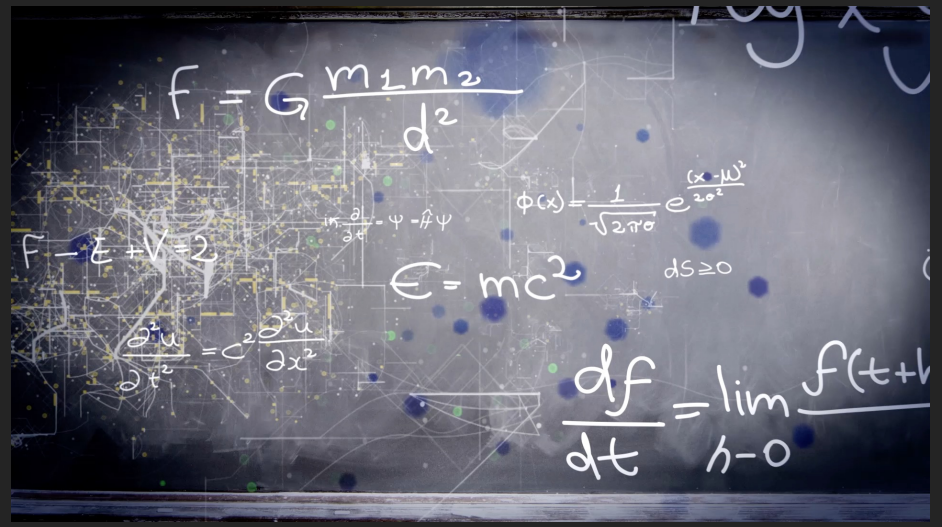
Machine Learning

Zheng Yimin / NSDAI -01 / P7053148



CLASSIFICATION : TITANIC DATASET

```
# shift-tab to show docstring: highlight and shift-tab: format
#?zip()
#%lsmagic
# Suppress Future Warnings
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

Data Importing

```
#importing all applicable libraries
import numpy as np
import pandas as pd
import seaborn as sns
import sklearn
import matplotlib
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC, LinearSVC
from sklearn import linear_model
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
from sklearn.model_selection import cross_val_score, GridSearchCV, cross_val_predict
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt
import platform

message="        Versions        "
print("*"*len(message))
print(message)
print("*"*len(message))
print("Scikit-learn version={}".format(sklearn.__version__))
print("Numpy version={}".format(np.__version__))
print("Pandas version={}".format(pd.__version__))
print("Matplotlib version={}".format(matplotlib.__version__))
print("Python version={}".format(platform.python_version()))
```

```
**************************
        Versions
**************************
Scikit-learn version=0.24.1
Numpy version=1.20.1
Pandas version=1.2.4
Matplotlib version=3.3.4
Python version=3.8.8
```

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

```
**************************************************
Summary of various features provided in the dataset
**************************************************
PassengerId: ID as generated in sequence in the dataset
Survived: Passengers whom survived = 1, Passengers whom did not survive = 0
Pclass: Passenger class in 1, 2 or 3
Name: Name of passenger
Sex: Gender of passenger
Age: Age of passenger
SibSp: Number of siblings or spouse onboard with the passenger
Parch: Number of parents or children onboard with the passenger
Ticket: Ticket number
Fare: Ticket fare paid by the passenger
Cabin: Cabin number
Embarked: Port embarked at - C, S or Q
```

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

Train Dataset

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Test Dataset

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	relatives	not_alone	Title
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q	0	1	1
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S	1	0	3
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q	0	1	1
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S	0	1	1
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S	2	0	3

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

```
print(traindf.info())
print(traindf.isnull().sum())
```

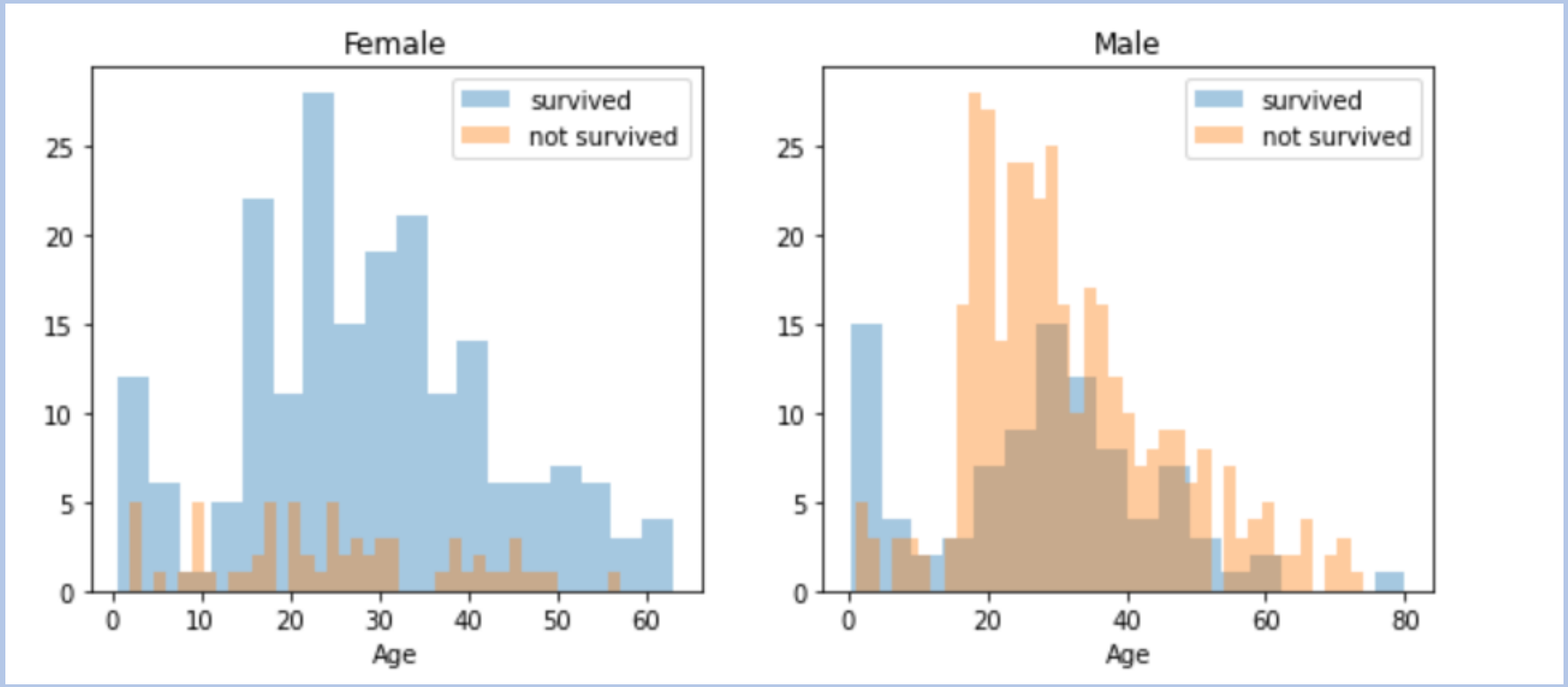
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
 11  relatives    418 non-null    int64
 12  not_alone    418 non-null    int64
 13  Title        418 non-null    int64
dtypes: float64(2), int64(7), object(5)
memory usage: 45.8+ KB
None
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
relatives        0
not_alone        0
Title            0
dtype: int64
```

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

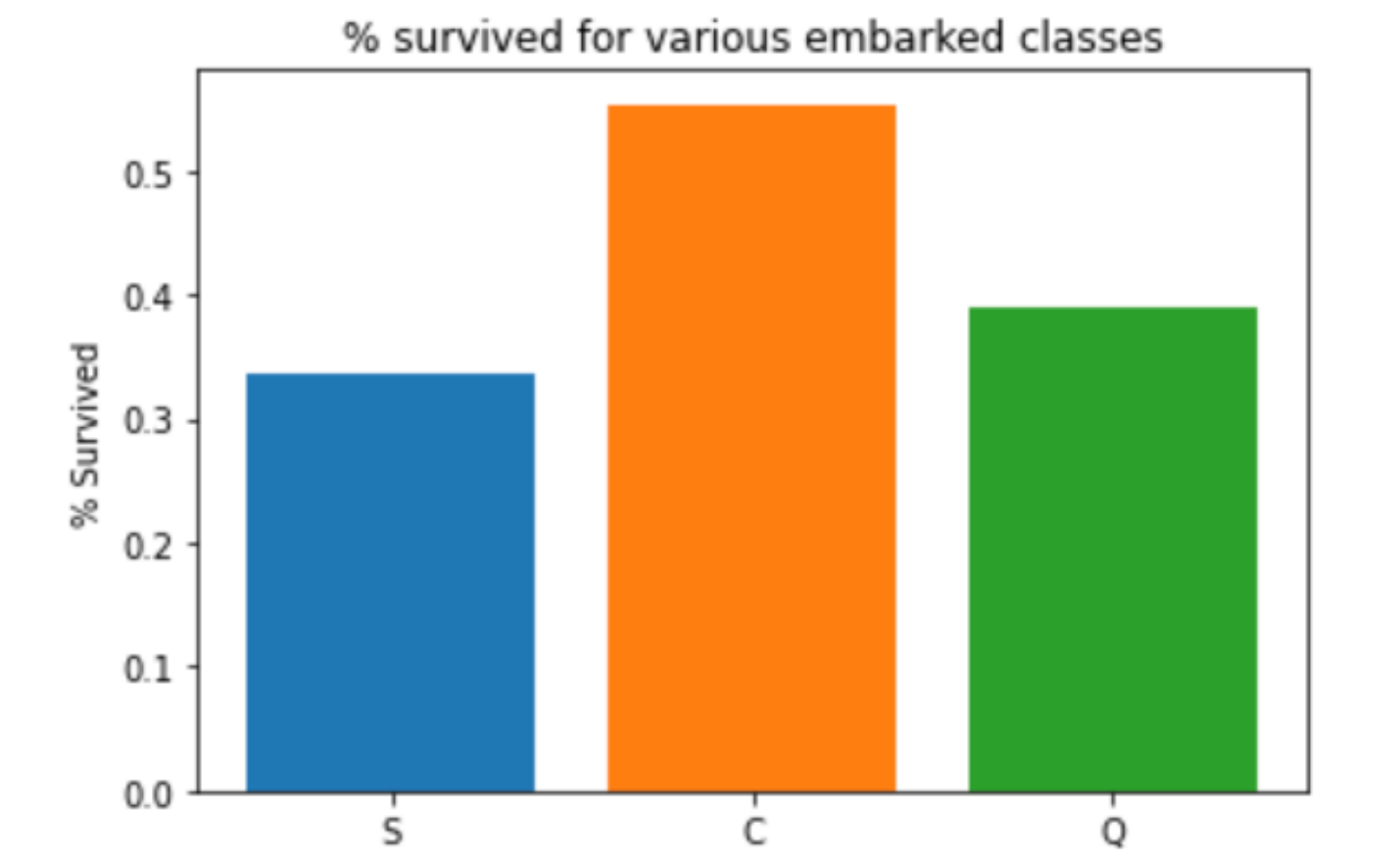
1. AGE & SEX



Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

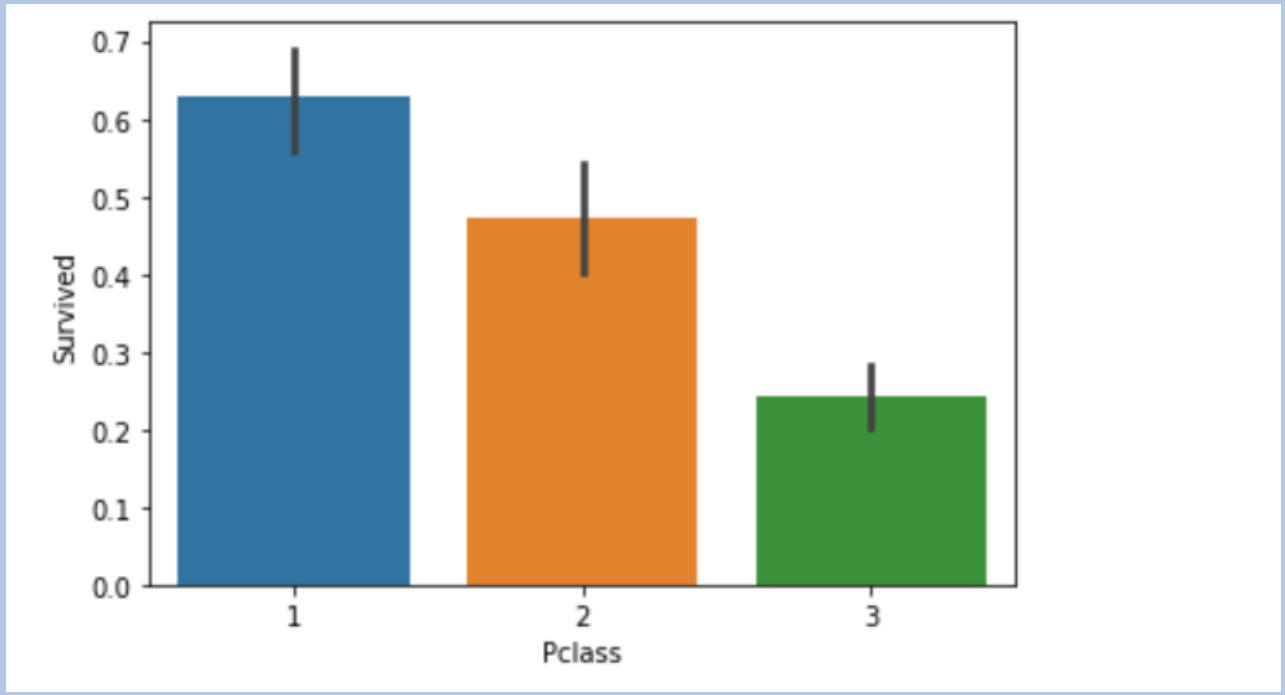
2. EMBARKED CLASS

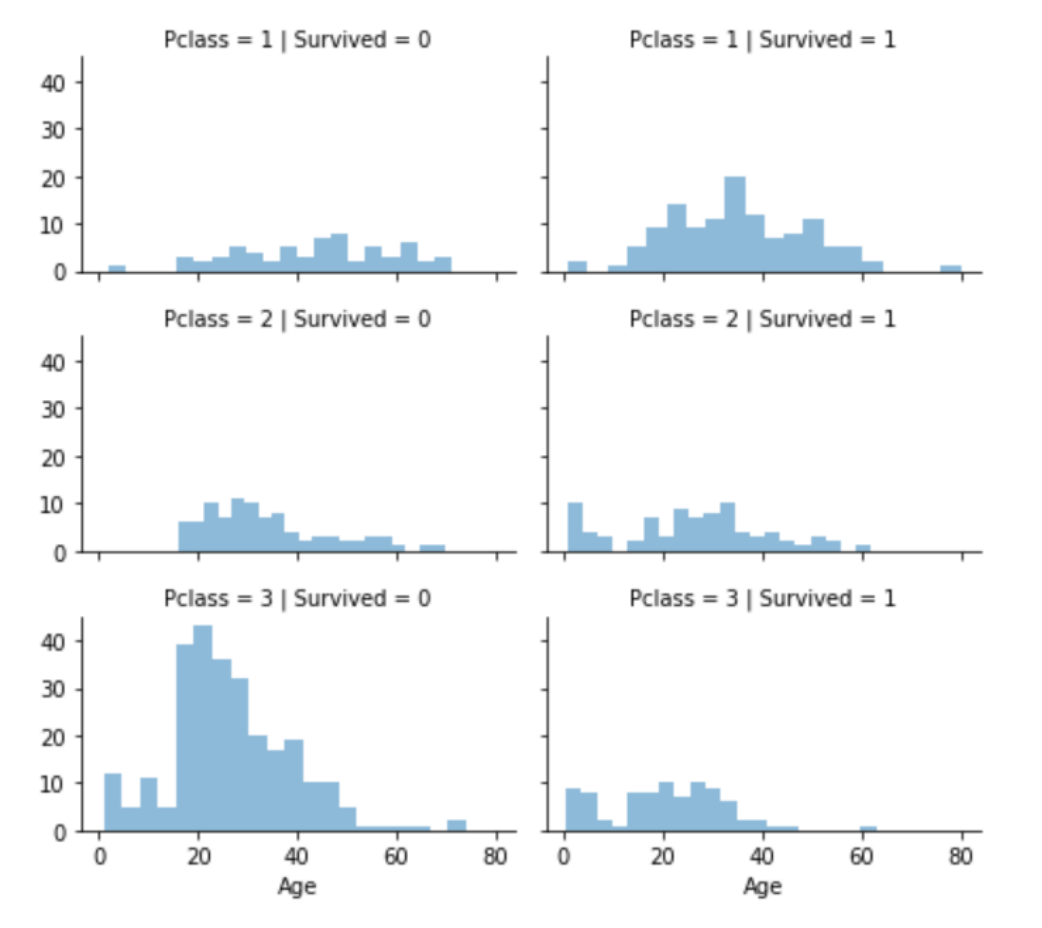


Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

3. PCLASS

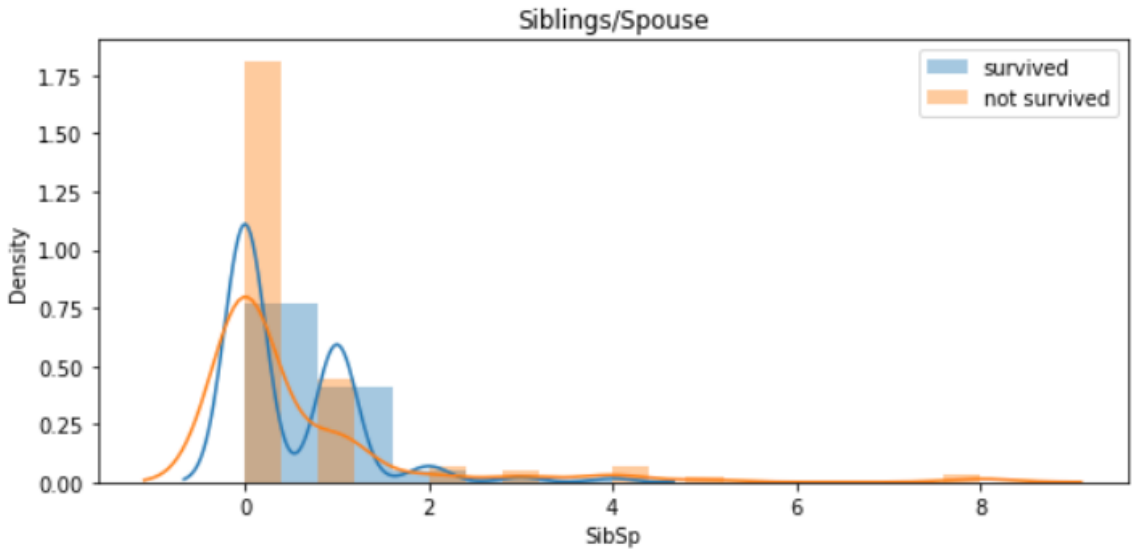




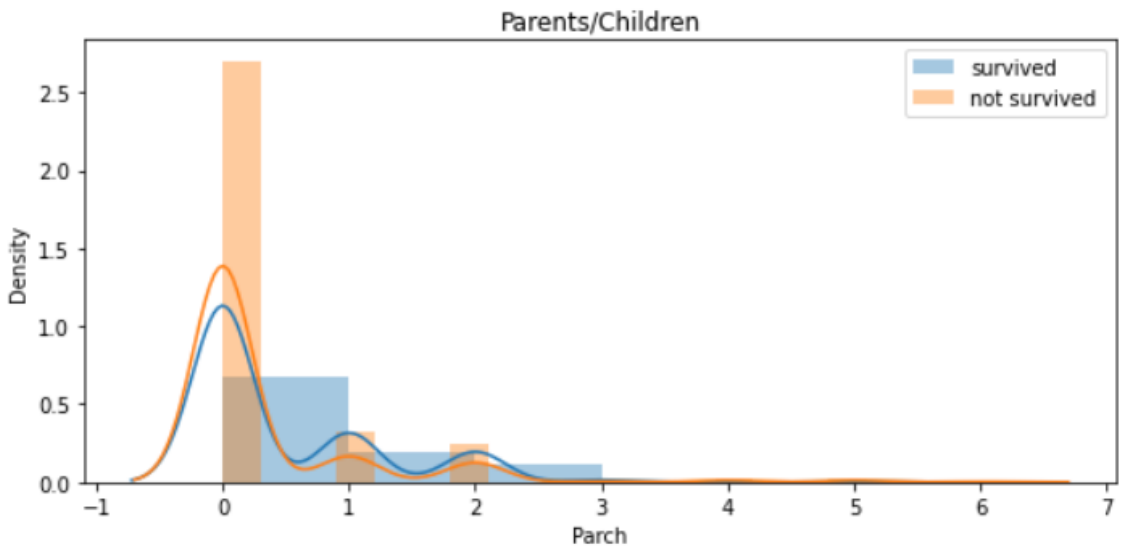
Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

4. SIBLINGS/SPOUSE



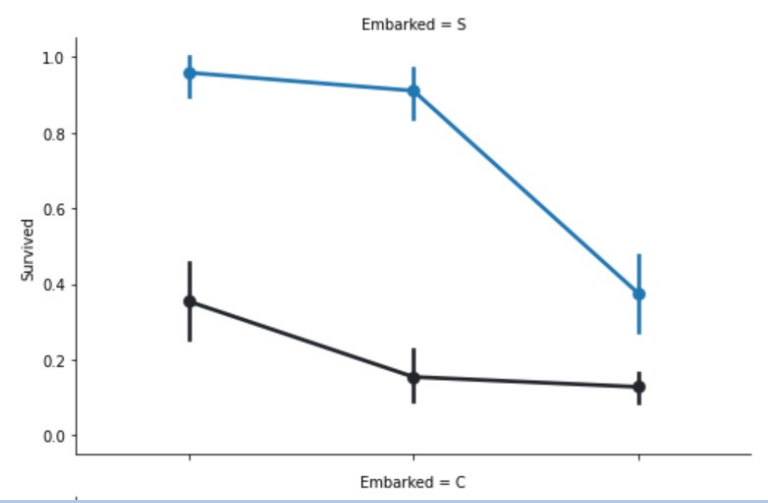
5. PARENTS/CHILDREN

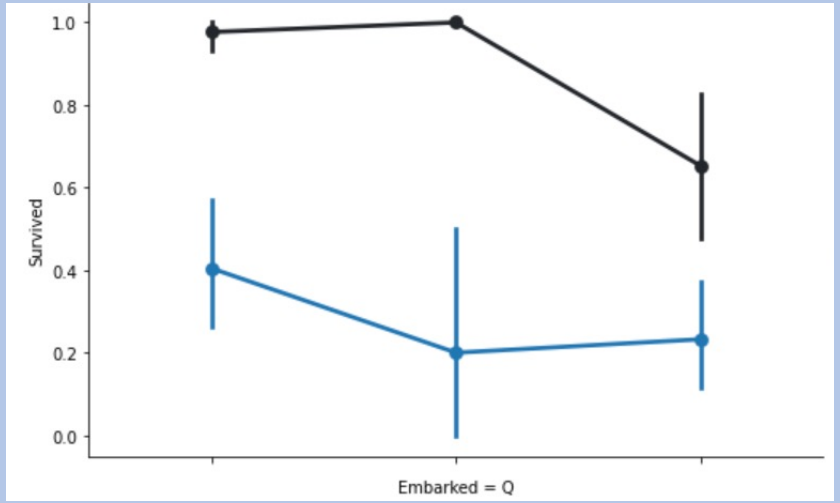


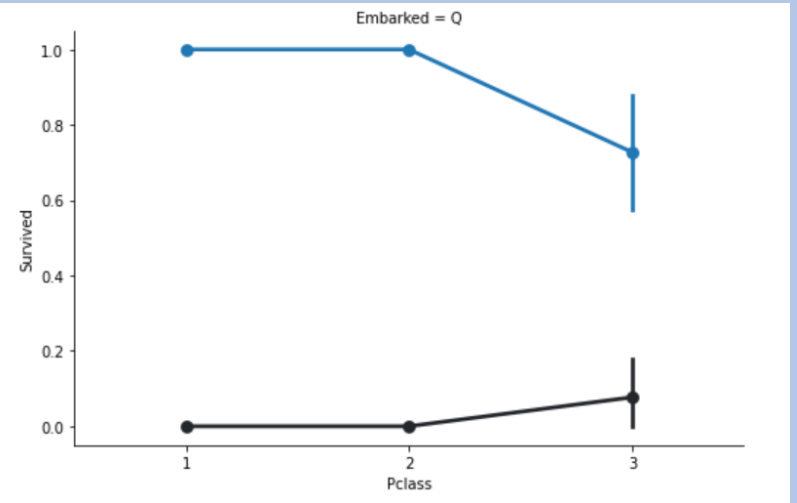
Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

6. PCLASS, EMBARKED AND SEX





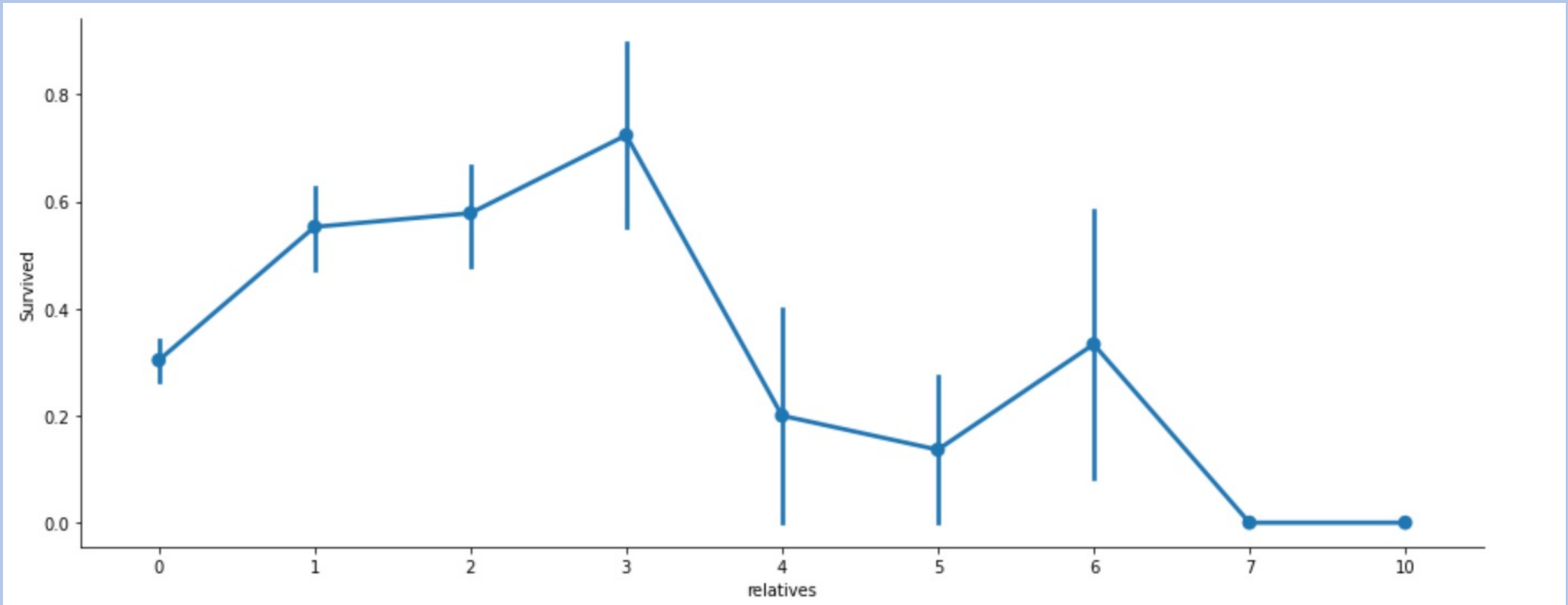


> MALE > FEMALE

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

7. RELATIVES – SIBSP + PARCH



CLASSIFICATION : TITANIC DATASET

```
#drop cabin column as it has too many null values
train_data = traindf.drop(columns='Cabin', axis=1)
test_data = testdf.drop(columns='Cabin', axis=1)

#Replacing the missing values in the "Age" column with the mean value
train_data['Age'].fillna(train_data['Age'].mean(), inplace=True)
test_data['Age'].fillna(test_data['Age'].mean(), inplace=True)
test_data['Fare'].fillna(test_data['Fare'].mean(), inplace=True)

#Replacing the missing values in the "Embarked" column with mode value
train_data['Embarked'].fillna(train_data['Embarked'].mode()[0], inplace=True)
test_data['Embarked'].fillna(test_data['Embarked'].mode()[0], inplace=True)

#Converting integer type values into categorical columns
train_data1=train_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}, inplace=True)
test_data1=test_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}, inplace=True)

#dropping these 3 columns as they have less relation to the predictions
test_data1 = test_data.drop(columns = ['PassengerId','Ticket', 'Name'],axis=1)
train_data1 = train_data.drop(columns = ['PassengerId','Ticket', 'Name'],axis=1)
```

Dropped:

- Cabin

Modified Null values for:

- Age
- Embarked
- Fare (test_data)

Replaced categories with values for:

- Sex
- Embarked

CLASSIFICATION : TITANIC DATASET

Train Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Survived    891 non-null    int64
     Pclass      891 non-null    int64
     Sex         891 non-null    int64
     Age         891 non-null    float64
 4   SibSp       891 non-null    int64
 5   Parch       891 non-null    int64
 6   Fare        891 non-null    float64
 7   Embarked    891 non-null    int64
 8   relatives   891 non-null    int64
 9   not_alone   891 non-null    int64
dtypes: float64(2), int64(8)
memory usage: 69.7 KB
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	relatives	not_alone
0	0	3	0	22.0	1	0	7.2500	0	1	0
1	1	1	1	38.0	1	0	71.2833	1	1	0
2	1	3	1	26.0	0	0	7.9250	0	0	1
3	1	1	1	35.0	1	0	53.1000	0	1	0
4	0	3	0	35.0	0	0	8.0500	0	0	1

Test Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Pclass      418 non-null    int64
 1   Sex         418 non-null    int64
 2   Age         418 non-null    float
 3   SibSp       418 non-null    int64
 4   Parch       418 non-null    int64
 5   Fare        418 non-null    float64
 6   Embarked    418 non-null    int64
 7   relatives   418 non-null    int64
 8   not_alone   418 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 29.5 KB
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	relatives	not_alone
0	3	0	34.5	0	0	7.8292	2	0	1
1	3	1	47.0	1	0	7.0000	0	1	0
2	2	0	62.0	0	0	9.6875	2	0	1
3	3	0	27.0	0	0	8.6625	0	0	1
4	3	1	22.0	1	1	12.2875	0	2	0

Import Data | EDA | **Data Preparation** | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

```
#Split the data into the target and feature variables

X = train_data1.drop(columns = ['Survived'],axis=1)
y = train_data1.Survived
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Models	Score - Training	Score - Test validation
Random Forest	0.980337	0.821229
Decision Tree	0.980337	0.782123
KNN	0.838483	0.703911
Logistic Regression	0.806180	0.804469
Naive Bayes	0.796348	0.787709
Linear SVC	0.766854	0.832402
Non-Linear SVC	0.679775	0.659218

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

feature	importance
Fare	0.270
Sex	0.266
Age	0.248
Pclass	0.074
relatives	0.052
Embarked	0.033
SibSp	0.026
Parch	0.021
not_alone	0.010

2. features_importances

1. Cross-validation score

```
#Cross validation with Random Forest
RF_cross_val = cross_val_score(RFmodel, X_train, y_train, cv=10, scoring = "accuracy")
print("Scores:", RF_cross_val)
print("Mean:", RF_cross_val.mean())
print("Standard Deviation:", RF_cross_val.std())
```

Scores: [0.80555556 0.76388889 0.73239437 0.83098592 0.76056338 0.77464789 0.77464789 0.78873239 0.81690141 0.87323944]
Mean: 0.7921557120500783
Standard Deviation: 0.038558442032076416

3. Confusion Matrix

```
array([[380,  64],
       [ 83, 185]])
```

Import Data | EDA | Data Preparation | Train Model | **Model Evaluation** | Model re-training

CLASSIFICATION : TITANIC DATASET

1. Dropping column(s) and reorganizing train / test(validation) data set

```
#dropping the less significant features
train_data_final = train_data1.drop(['not_alone'], axis=1)
test_data_final = test_data1.drop(['not_alone'], axis=1)

X1 = train_data_final.drop(columns = ['Survived'],axis=1)
y1 = train_data_final.Survived

X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(X1, y1, test_size=0.2, random_state=42)
```

2. Re-training Random Forest model

```
#Random forest - new
RFmodel_new = RandomForestClassifier(n_estimators=100)

RFmodel_new.fit(X_train_new, y_train_new)

#Checking accuracy
X_train_predictionRFnew = RFmodel_new.predict(X_train_new)

training_data_accuracyRFnew = accuracy_score(y_train_new, X_train_predictionRFnew)
print('Accuracy score of training data : ', training_data_accuracyRFnew)

#predicting the accuracy score with test data
Y_predRF_new = RFmodel_new.predict(X_test_new)

test_data_accuracyRFnew = accuracy_score(y_test_new, Y_predRF_new)
print('Accuracy score of test data : ', test_data_accuracyRFnew)
```

CLASSIFICATION : TITANIC DATASET

```
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}

# Instantiate the grid search model
grid_search = GridSearchCV(estimator = RFmodel_new, param_grid = param_grid, cv = 3, n_jobs = -1, verbose = 2)
```

```
# Fit the grid search to the data
grid_search.fit(X_train_new, y_train_new)
grid_search.best_params_
```

```
Fitting 3 folds for each of 288 candidates, totalling 864 fits

{'bootstrap': True,
 'max_depth': 110,
 'max_features': 3,
 'min_samples_leaf': 5,
 'min_samples_split': 8,
 'n_estimators': 100}
```

3. Hyperparameter tuning

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET

1. Cross-validation score

```
#Cross validation with Random Forest
RF_cross_val1 = cross_val_score(random_forest, X_train_new, y_train_new, cv=10, scoring = "accuracy")
print("Scores:", RF_cross_val1)
print("Mean:", RF_cross_val1.mean())
print("Standard Deviation:", RF_cross_val1.std())
```

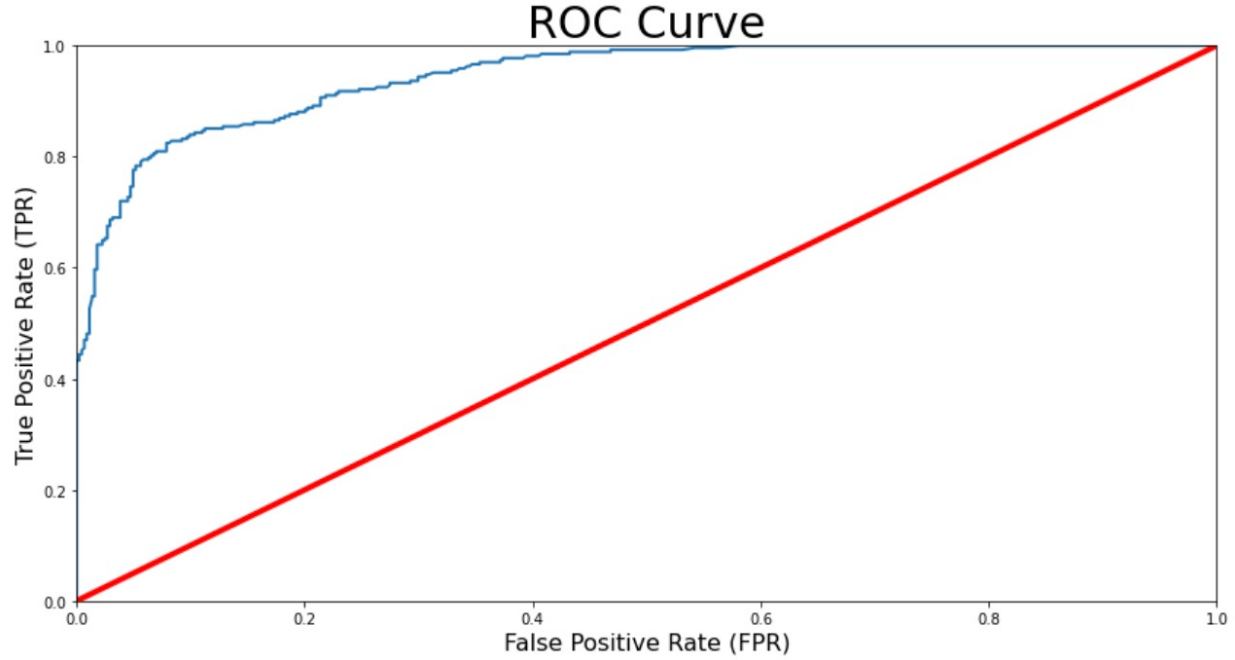
Scores: [0.86111111 0.81944444 0.76056338 0.97183099 0.83098592 0.76056338 0.83098592 0.76056338 0.74647887 0.90140845]
Mean: 0.8243935837245697
Standard Deviation: 0.0688638166954511 2

2. Confusion Matrix

```
array([[405,  39],
       [ 88, 180]])
```

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

CLASSIFICATION : TITANIC DATASET



ROC-AUC Score = 94.44%

Models	r^2 Score - Training	r^2 Score - Test
Random Forest	0.869382	0.826816
Baseline Model - most_frequent	0.623596	0.586592
Baseline Model - Constant	0.376404	0.413408

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

REGRESSION: HOUSING PRICES DATASET

```
# shift-tab to show docstring: highlight and shift-tab: format
#?zip()
#%lsmagic
# Suppress Future Warnings
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
#importing all applicable libraries
import numpy as np
import pandas as pd
import seaborn as sns
import sklearn
import matplotlib
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVR
from sklearn import linear_model, ensemble
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score, r2_score, mean_squared_error, m
from sklearn.model_selection import cross_val_score, GridSearchCV, cross_val_predict
from sklearn.dummy import DummyRegressor
import matplotlib.pyplot as plt
import platform

message="        Versions        "
print("*"*len(message))
print(message)
print("*"*len(message))
print("Scikit-learn version={}".format(sklearn.__version__))
print("Numpy version={}".format(np.__version__))
print("Pandas version={}".format(pd.__version__))
print("Matplotlib version={}".format(matplotlib.__version__))
print("Python version={}".format(platform.python_version()))
```

```
**************************
        Versions
**************************
Scikit-learn version=0.24.1
Numpy version=1.20.1
Pandas version=1.2.4
Matplotlib version=3.3.4
Python version=3.8.8
```

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

REGRESSION: HOUSING PRICES DATASET

id - Unique ID for each home sold
date - Date of the home sale
price - Price of each home sold
bedrooms - Number of bedrooms
bathrooms - Number of bathrooms, where .5 accounts for a room with a toilet but no shower
sqft_living - Square footage of the apartments interior living space
sqft_lot - Square footage of the land space
floors - Number of floors
waterfront - A dummy variable for whether the apartment was overlooking the waterfront or not
view - An index from 0 to 4 of how good the view of the property was
condition - An index from 1 to 5 on the condition of the apartment,
grade - An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design.
sqft_above - The square footage of the interior housing space that is above ground level
sqft_basement - The square footage of the interior housing space that is below ground level
yr_built - The year the house was initially built
yr_renovated - The year of the house's last renovation
zipcode - What zipcode area the house is in
lat - Lattitude
long - Longitude
sqft_living15 - The square footage of interior housing living space for the nearest 15 neighbors
sqft_lot15 - The square footage of the land lots of the nearest 15 neighbors

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21613 non-null  int64
 1   date           21613 non-null  object
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64
 20  sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
None
```

Import Data | **EDA** | Data Preparation | Train Model | Model Evaluation | Model re-training

REGRESSION: HOUSING PRICES DATASET

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987

5 rows × 21 columns

	id	price	bedrooms	bathrooms	sqft_living
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000

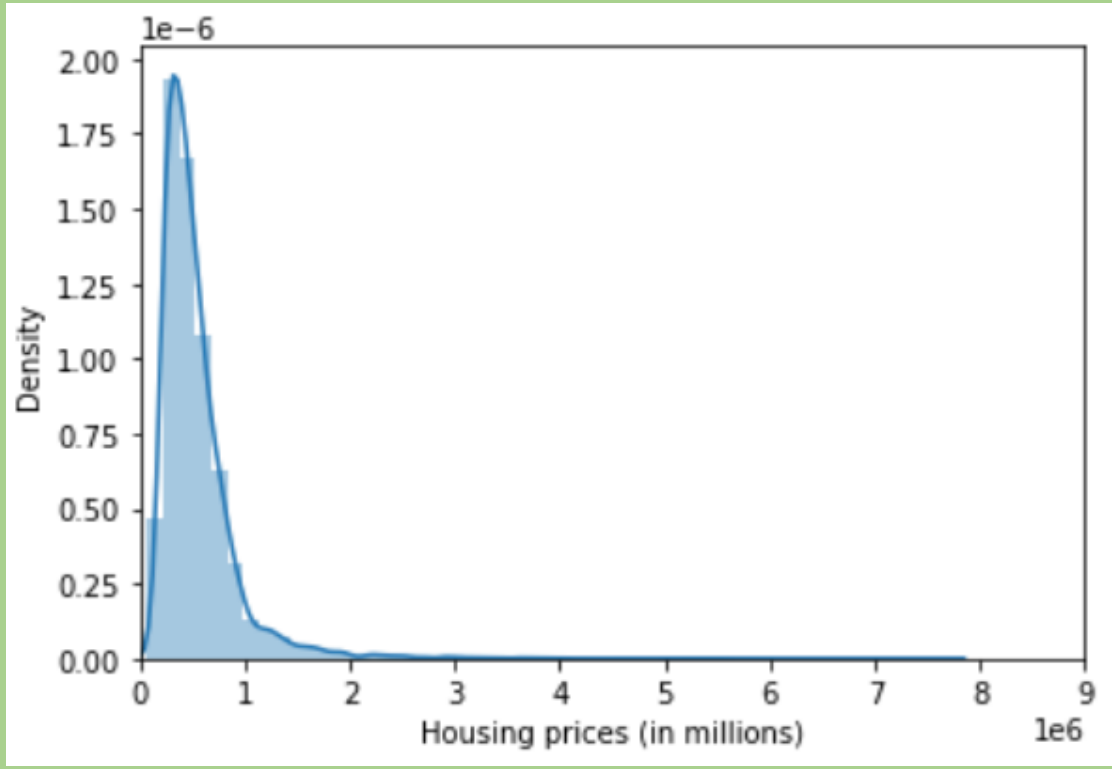
	sqft_lot	floors	waterfront	view	condition
count	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000
mean	1.510697e+04	1.494309	0.007542	0.234303	3.409430
std	4.142051e+04	0.539989	0.086517	0.766318	0.650743
min	5.200000e+02	1.000000	0.000000	0.000000	1.000000
25%	5.040000e+03	1.000000	0.000000	0.000000	3.000000
50%	7.618000e+03	1.500000	0.000000	0.000000	3.000000
75%	1.068800e+04	2.000000	0.000000	0.000000	4.000000
max	1.651359e+06	3.500000	1.000000	4.000000	5.000000

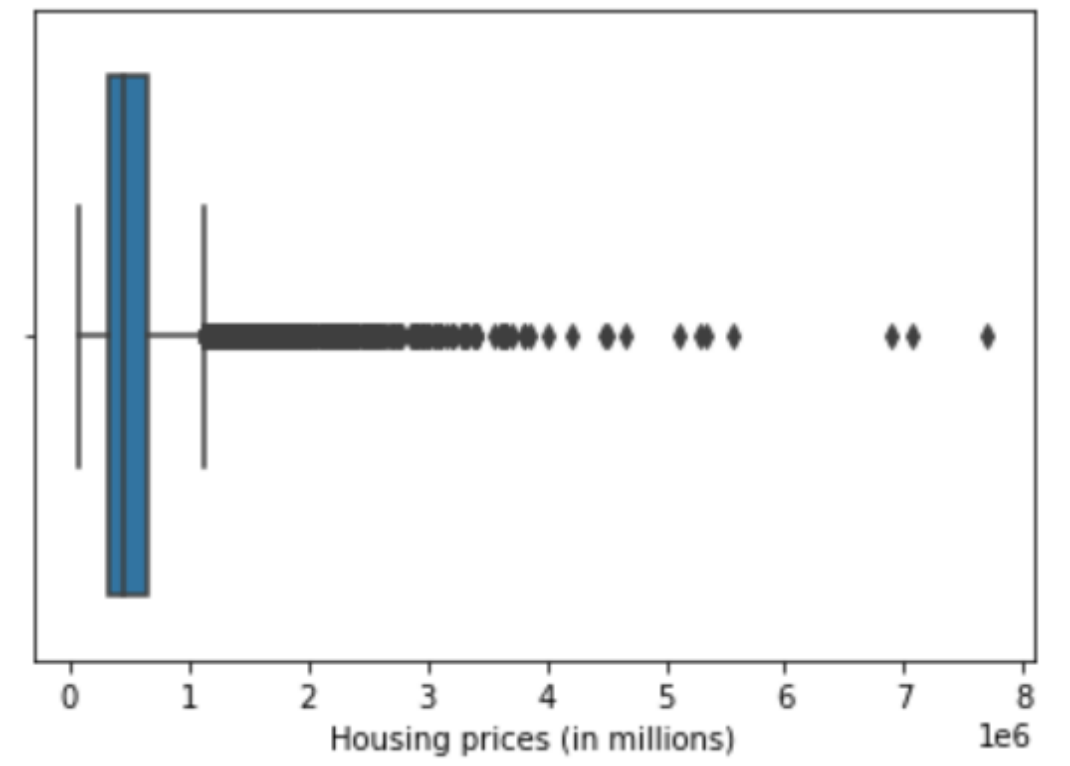
	grade	sqft_above	sqft_basement	yr_built	yr_renovated
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	7.656873	1788.390691	291.509045	1971.005136	84.402258
std	1.175459	828.090978	442.575043	29.373411	401.679240
min	1.000000	290.000000	0.000000	1900.000000	0.000000
25%	7.000000	1190.000000	0.000000	1951.000000	0.000000
50%	7.000000	1560.000000	0.000000	1975.000000	0.000000
75%	8.000000	2210.000000	560.000000	1997.000000	0.000000
max	13.000000	9410.000000	4820.000000	2015.000000	2015.000000

	zipcode	lat	long	sqft_living15	sqft_lot15
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	98077.939805	47.560053	-122.213896	1986.552492	12768.455652
std	53.505026	0.138564	0.140828	685.391304	27304.179631
min	98001.000000	47.155900	-122.519000	399.000000	651.000000
25%	98033.000000	47.471000	-122.328000	1490.000000	5100.000000
50%	98065.000000	47.571800	-122.230000	1840.000000	7620.000000
75%	98118.000000	47.678000	-122.125000	2360.000000	10083.000000
max	98199.000000	47.777600	-121.315000	6210.000000	871200.000000

Import Data | **EDA** | Data Preparation | Train Model | Model Evaluation | Model re-training

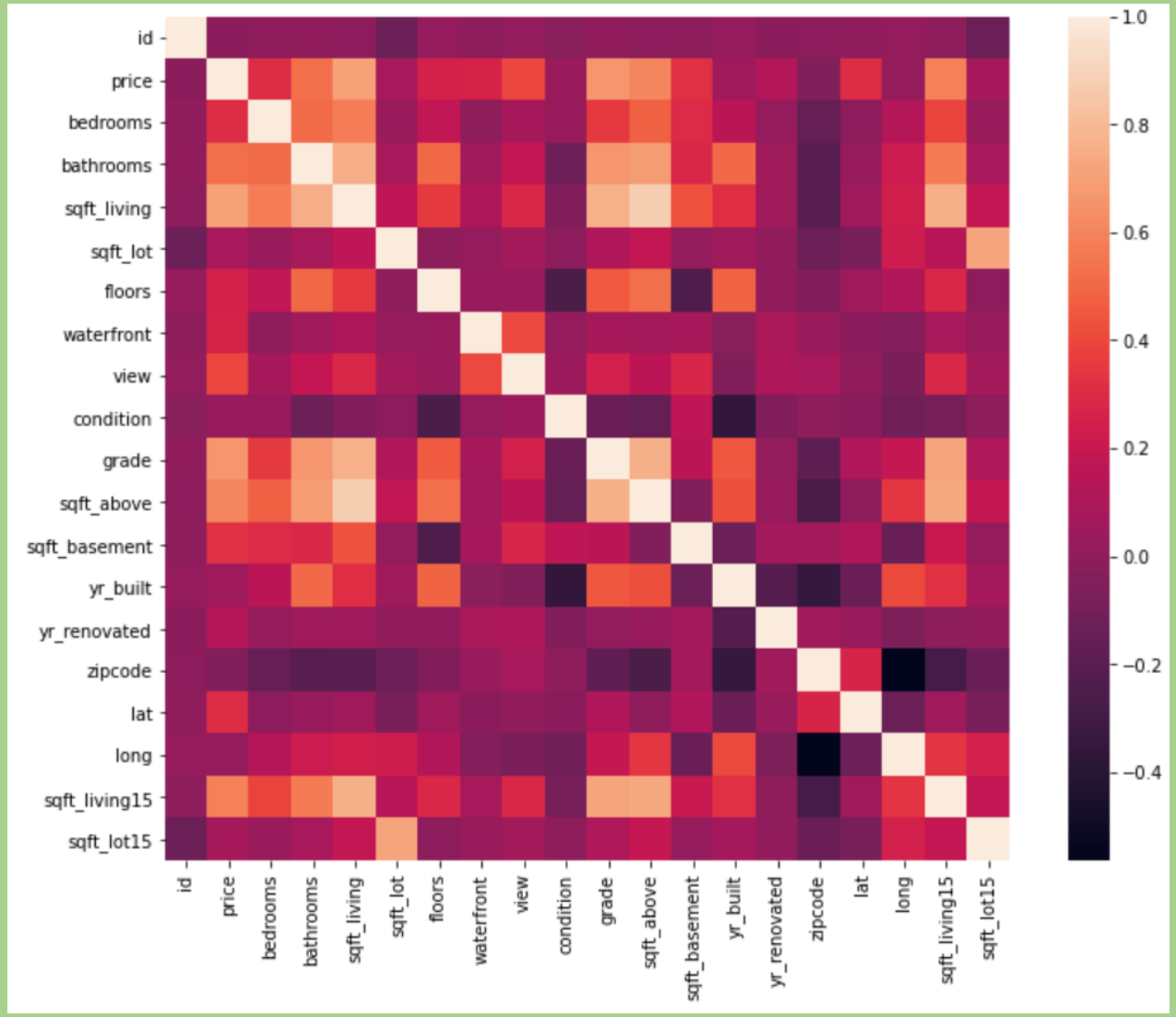
REGRESSION: HOUSING PRICES DATASET





Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

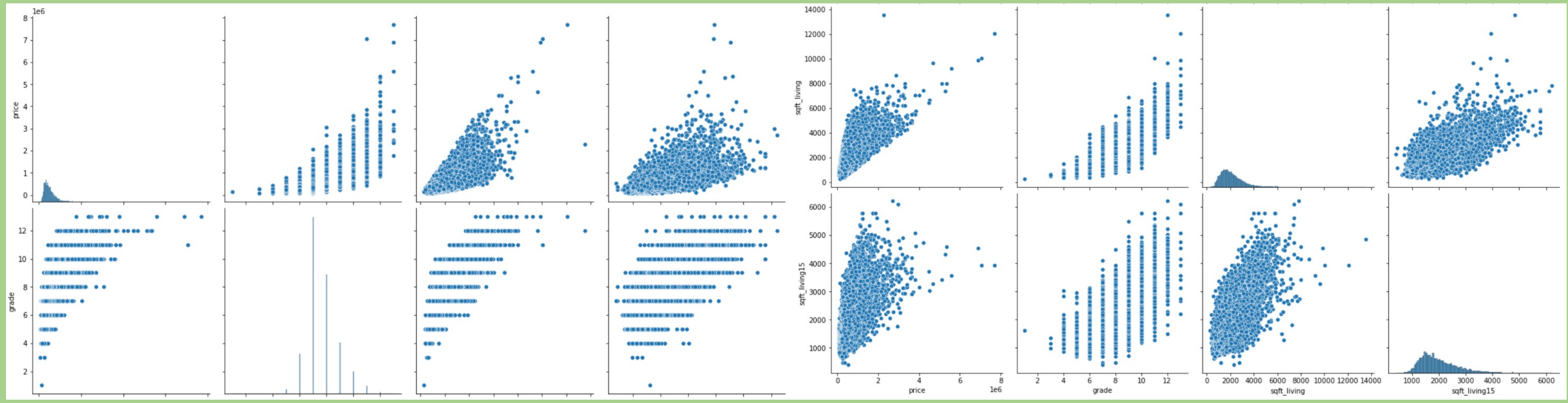
REGRESSION: HOUSING PRICES DATASET



Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

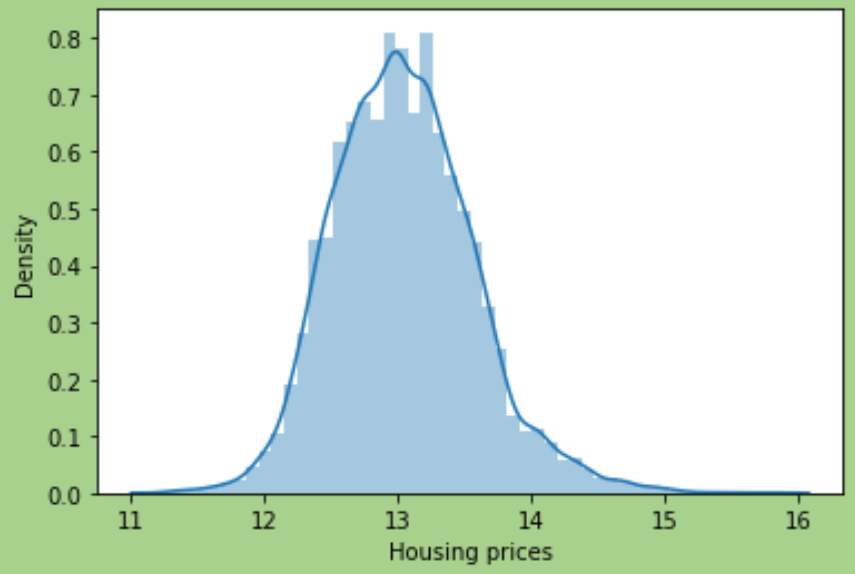
REGRESSION: HOUSING PRICES DATASET

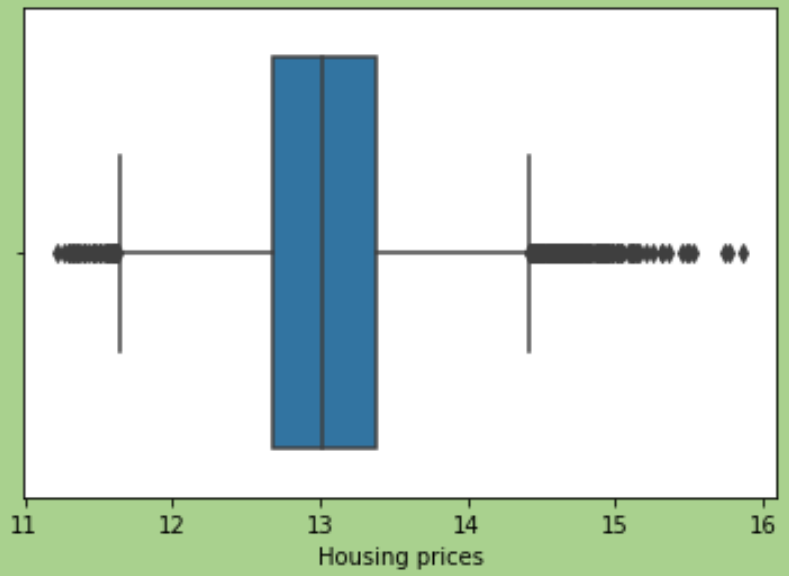
Top 4 correlated features



Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

REGRESSION: HOUSING PRICES DATASET





Added:

- *Numpy_log for prices of houses*

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

REGRESSION: HOUSING PRICES DATASET

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20467 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             20467 non-null  int64
 1   date           20467 non-null  object
 2   price          20467 non-null  float64
 3   bedrooms       20467 non-null  int64
 4   bathrooms      20467 non-null  float64
 5   sqft_living    20467 non-null  int64
 6   sqft_lot       20467 non-null  int64
 7   floors         20467 non-null  float64
 8   waterfront     20467 non-null  int64
 9   view           20467 non-null  int64
 10  condition      20467 non-null  int64
 11  grade          20467 non-null  int64
 12  sqft_above     20467 non-null  int64
 13  sqft_basement  20467 non-null  int64
 14  yr_built       20467 non-null  int64
 15  yr_renovated   20467 non-null  int64
 16  zipcode        20467 non-null  int64
 17  lat            20467 non-null  float64
 18  long           20467 non-null  float64
 19  sqft_living15  20467 non-null  int64
 20  sqft_lot15     20467 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.4+ MB
```

	0
id	0
date	0
price	0
bedrooms	12
bathrooms	9
sqft_living	0
sqft_lot	0
floors	0
waterfront	20406
view	18884
condition	0
grade	0
sqft_above	0
sqft_basement	12702
yr_built	0
yr_renovated	19701
zipcode	0
lat	0
long	0
sqft_living15	0
sqft_lot15	0

Dropped:

- *Outliers for sqft_living + price*
- *Waterfront*
- *View*
- *Sqft_basement*
- *Yr_renovated*

Import Data | EDA | **Data Preparation** | Train Model | Model Evaluation | Model re-training

REGRESSION: HOUSING PRICES DATASET

```
#Using Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

```
#Using Decision Tree Regression
dt = DecisionTreeRegressor(random_state = 0)
dt.fit(X_train, y_train)
y_pred1 = dt.predict(X_test)
```

```
#Using Support Vector Regression
svr = SVR(kernel = 'rbf')
svr.fit(X_train, y_train)
y_pred2 = svr.predict(X_test)
```

```
#Using KNN
knn = KNeighborsRegressor(n_neighbors = 3)
knn.fit(X_train, y_train)
y_pred3 = knn.predict(X_test)
```

```
#Using Ensemble Gradient Boosting Regressor - Model 1
ens = ensemble.GradientBoostingRegressor(n_estimators = 400,
                                         max_depth = 5,
                                         min_samples_split = 2,
                                         learning_rate = 0.1,
                                         loss = 'ls')
ens.fit(X_train, y_train)
y_pred4 = ens.predict(X_test)
```

REGRESSION: HOUSING PRICES DATASET

Models	r^2 Score - Training	r^2 Score - Test
Decision Tree	0.996784	0.766580
Gradient Boosting Regressor	0.943002	0.900503
Linear Regression	0.756372	0.752465
KNeighborsRegressor	0.737702	0.462408
SVR	0.513042	0.514959

REGRESSION: HOUSING PRICES DATASET

```
#Because of the consistency of the prediction scores and the high results produced for both test and training sets as p

#Hyperparameter tuning 1 - Gradient Boosting Regressor Model 2
ens1 = ensemble.GradientBoostingRegressor(n_estimators = 1000,
                                          max_depth = 5,
                                          min_samples_split = 2,
                                          learning_rate = 0.1,
                                          loss = 'ls')
ens1.fit(X_train, y_train)
y_pred4 = ens1.predict(X_test)
```

```
#Hyperparameter tuning 1 - Gradient Boosting Regressor Model 3
ens2 = ensemble.GradientBoostingRegressor(n_estimators = 1000,
                                          max_depth = 7,
                                          min_samples_split = 2,
                                          learning_rate = 0.1,
                                          loss = 'ls')
ens2.fit(X_train, y_train)
y_pred5 = ens2.predict(X_test)
```

```
#Hyperparameter tuning 1 - Gradient Boosting Regressor Model 4
ens3 = ensemble.GradientBoostingRegressor(n_estimators = 1000,
                                          max_depth = 5,
                                          min_samples_split = 500,
                                          learning_rate = 0.1,
                                          loss = 'ls')
ens3.fit(X_train, y_train)
y_pred6 = ens3.predict(X_test)
```

Hyperparameter tuning

Gradient Boosting Regressor Models	r^2 Score - Training	r^2 Score - Test
Model 1	0.943002	0.900503
Model 2	0.967676	0.899625
Model 3	0.991261	0.894137
Model 4	0.936692	0.900044

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training

REGRESSION: HOUSING PRICES DATASET

Models	r^2 Score - Training	r^2 Score - Test
Gradient Boosting Regressor - Model 1	0.943002	0.900503
Baseline Model - Mean	0.000000	-0.000004
Baseline Model - Constant	-27256.386312	-27721.490471

Import Data | EDA | Data Preparation | Train Model | Model Evaluation | Model re-training