In [ ]:
```
import numpy as np
from numpy import arange
import pandas as pd
from pandas import set_option
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.preprocessing import StandardScaler
import os
import re
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
path="C:\\Users\\moham\\Documents\\JNTU_DataScience\\JNTU_machine_learning\\ml_assignments\\M

df=pd.read_csv(path)
```

In [ ]:
```
df.info()
```

In [ ]:
```
df.head(5)
```

In [ ]:
```
breadcrumb={'Buy>NT>DARWIN CITY':1,'Buy>NT>DARWIN':2}
df['breadcrumb']=df['breadcrumb'].replace(breadcrumb)

category_name={'Real Estate & Property for sale in DARWIN CITY, NT 0800':1,'Real Estate & Pro
df['category_name']=df['category_name'].replace(category_name)
```

In [ ]:
```
print(df['breadcrumb'],df['category_name'])
```

In [ ]:
```
# Import label encoder
from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
# Encode labels in column 'Country'.
df['property_type']= label_encoder.fit_transform(df['property_type'])
df['building_size']=label_encoder.fit_transform(df['building_size'])
df['land_size']=label_encoder.fit_transform(df['land_size'])
df['location_type']=label_encoder.fit_transform(df['location_type'])
df['city']=label_encoder.fit_transform(df['city'])
df['price']=label_encoder.fit_transform(df['price'])
```

```
#print(df.head())
```

In [ ]:
```
df.info()
```

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

In [ ]:
```
#df['price'] = pd.to_numeric(df['price'], errors='coerce')
```

In [ ]:
```
df['price']
```

In [ ]:
```
df['TID']
```

In [ ]:
```
df_Xtrain = df.iloc[:, [2,4,5,12,16,23,24]]
```

In [ ]:
```
df_Ytrain=df.iloc[:,[10]]

validation_size=0.20
seed=7
```

In [ ]:
```
df_Xtrain
```

In [ ]:
```
df_Xtrain.isna().sum()
```

In [ ]:
```
df_Ytrain.isna().sum()
```

In [ ]:
```
df_Xtrain.isna().mean()
```

In [ ]:
```
df_Xtrain['bedroom_count']
```

In [ ]:
```
#df_Xtrain.fillna(df_Xtrain['bedroom_count'].mean(), inplace=True)
#df_Xtrain = df_Xtrain[np.isfinite(df_Xtrain).all(1)]
df_Xtrain.fillna(df_Xtrain.mean(), inplace=True)
```

In [ ]:
```
df_Xtrain.describe()
```

In [ ]:
```
sns.displot(df_Xtrain['property_type'])
```

In [ ]:
```
sns.displot(df_Xtrain['building_size'])
```

In [ ]:
```
df_Xtrain.hist(sharex=False,sharey=False,xlabelsize=1,ylabelsize=1)
plt.show()
```

In [ ]:
```
df_Xtrain.plot(kind='density',subplots=True,layout=(4,4),sharex=False,legend=False,fontsize=1
plt.show()
```

In [ ]:
```
X_train,X_validation,Y_train,Y_validation=train_test_split(df_Xtrain,df_Ytrain,test_size=vali
```

In [ ]:
```
models=[]
#names=[]

models.append(('LR',LogisticRegression()))
models.append(('KNN',KNeighborsRegressor()))
models.append(('CART',DecisionTreeRegressor()))
models.append(('SVR',SVR()))

results=[]
names=[]
for name , model in models:
    kfold=KFold(n_splits=10,shuffle=True,random_state=seed)
    cv_results=cross_val_score(model,X_train,Y_train,cv=kfold,scoring='mean_squared_error')
```

```
    results.append(cv_results)
    names.append(name)

    msg="%s : %f  (%f)" % (name, cv_results.mean(),cv_results.std())
    print(msg)
```

In [ ]:
```
reg = linear_model.LinearRegression()
```

In [ ]:
```
reg.fit(X_train, Y_train)
```

In [ ]:
```
reg.coef_
```

In [ ]:
```
reg.intercept_
```

In [110…
```
y_pred = reg.predict(X_validation)
MSEValue = mean_squared_error(Y_validation, y_pred, multioutput='raw_values')
print('Mean Squared Error Value is : ', np.sqrt(MSEValue))
```

```
Mean Squared Error Value is :  [149.02664056]
```

In [ ]: