Machine Learning Model for Real Estate

Objective

Machine Learning in real estate offers great advantages, including better leads, quick operations, recued costs, etc. The investors get some excellent benefits such as better ROI, profitable business deals, better competitive advantage, automated operations, and increased employee efficiency. So by using machine learning model we can predict the clusters about hosue price predictions.

The dataset is collected from Kaggle. It contains data about:

1. Transaction date
2. House age
3. Distance to the nearest MRT station
4. Number of convenience stores
5. Latitude X6 longitude
6. House price of unit area

Link of the data --> "https://www.kaggle.com/datasets/quantbruce/real-estate-price-prediction?select=Real+estate.csv" (https://www.kaggle.com/datasets/quantbruce/real-estate-price-prediction?select=Real+estate.csv")

Machine Learning Models

In this report we will use three machine learning models and select the best performing model for prediction, because our objective is to predict the premium charges which is continuos so we will use regression models of machine learning. Below are the machine learning model we will use:

1. Support Vector Regressor (SVR)
2. K-Means Clustering
3. K-Means + SVR Implementation

Support Vector Regression uses the same principle behind Support Vector Machine. SVR also builds a hyperplane in an N-Dimensional vector space, where N is the number of features involved. Instead of keeping data points away from the hyperplane, here we aim to keep data points inside the hyperplane for regression. One of the tunable parameters is ε (epsilon), which is the width of the tube I create in the feature space. The next regularization parameter that we hypertune is C which controls the "slack" (ξ) that measures the distance to points outside the tube.

Using python libraries sklearn, numpy, pandas and matplotlib I created the model and visualize it's outcome. Below is the code to go through for further understanding.

Key Findings

Regression is one of the straightforward tasks carried out in Machine Learning. Still, it can become increasingly difficult to find the right algorithm that can build a model with decent accuracy for deploying to production. It is always better to try most of the algorithms out there before finalizing one. There may be instances where a simple linear regression would have worked wonders instead of the huge random forest regression. Having a solid understanding of the data and its nature will help us choose algorithms easily and visualization also helps in the process. In this project I only focus on exploring the possibility of the combination of K-Means clustering and Support Vector Regression.

Action plan

This approach won't always suit all the cases of regression as it stresses the clustering possibility. One good example of such a problem can be soil moisture prediction as it may vary with seasons and a lot of cyclic features, resulting in clustered data points. K-Means clustering saves us from manual decisions on clusters and SVR handles non-linearity very well. Both SVR and K-Means alone have a lot more in-depth to explore including Kernels, centroid initialization, and the list goes on.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.svm import SVC, SVR #for building support vector classification and regression model
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.model_selection import train_test_split

import statistics
from scipy import stats

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go # for data visualization
import plotly.express as px # for data visualization

%matplotlib inline
```

In [2]:
```
# Read data into a dataframe
# https://www.kaggle.com/datasets/quantbruce/real-estate-price-prediction?select=Real+estate.csv

df = pd.read_csv('Real estate.csv', encoding='utf-8')
X=df['X2 house age'].values.reshape(-1,1)
y=df['Y house price of unit area'].values

df.head()
```

Out[2]:

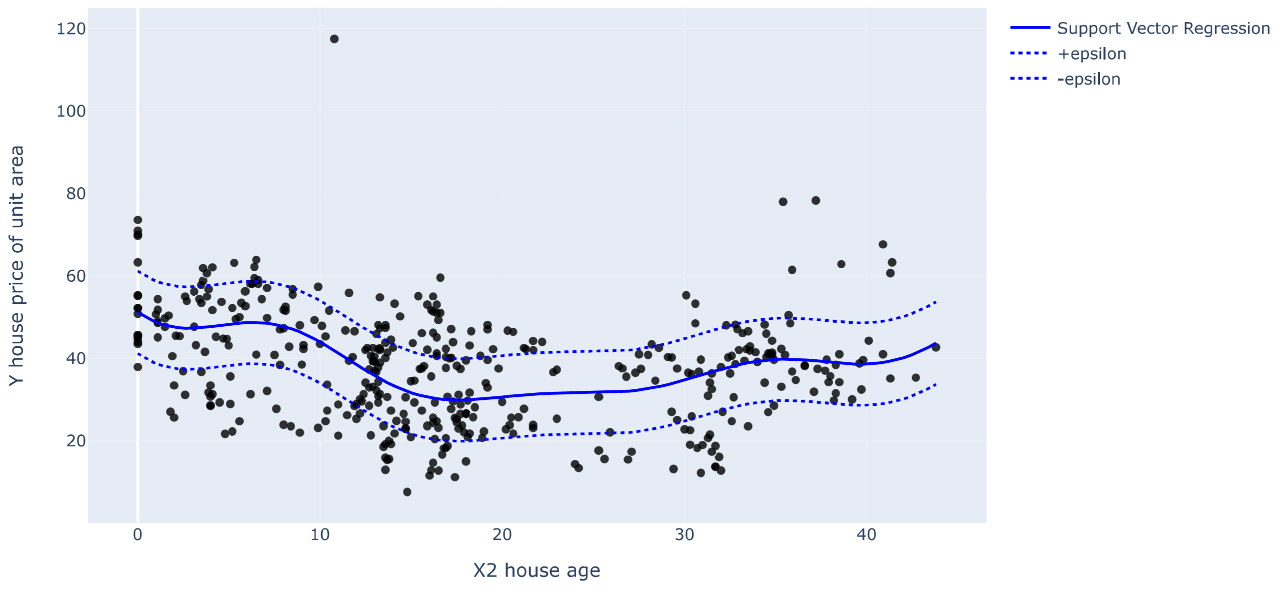
	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

SVR

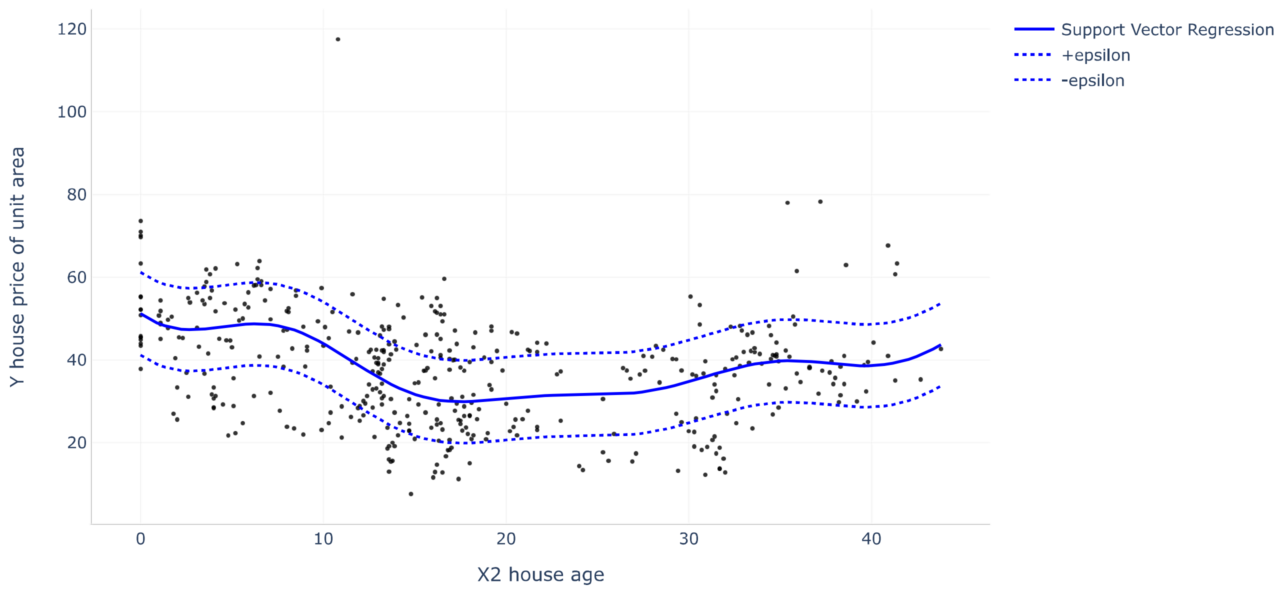
In [3]:
```
# Building SVR Model
model = SVR(kernel='rbf', C=1000, epsilon=1) # set kernel and hyperparameters
svr = model.fit(X, y)
```

In [4]:
```
x_range = np.linspace(X.min(), X.max(), 100)
y_svr = model.predict(x_range.reshape(-1, 1))

plt = px.scatter(df, x=df['X2 house age'], y=df['Y house price of unit area'],
                 opacity=0.8, color_discrete_sequence=['black'])
```

In [5]:
```
# Add SVR Hyperplane
plt.add_traces(go.Scatter(x=x_range,
    y=y_svr, name='Support Vector Regression', line=dict(color='blue')))
plt.add_traces(go.Scatter(x=x_range,
    y=y_svr+10, name='+epsilon', line=dict(color='blue', dash='dot')))
plt.add_traces(go.Scatter(x=x_range,
    y=y_svr-10, name='-epsilon', line=dict(color='blue', dash='dot')))
```



In [6]:
```
# Set chart background color
plt.update_layout(dict(plot_bgcolor = 'white'))

# Updating axes lines
plt.update_xaxes(showgrid=True, gridwidth=1, gridcolor='lightgrey',
                 zeroline=True, zerolinewidth=1, zerolinecolor='lightgrey',
                 showline=True, linewidth=1, linecolor='black')

plt.update_yaxes(showgrid=True, gridwidth=1, gridcolor='lightgrey',
                 zeroline=True, zerolinewidth=1, zerolinecolor='lightgrey',
                 showline=True, linewidth=1, linecolor='black')

# Update marker size
plt.update_traces(marker=dict(size=3))

plt.show()
```



K-Means Clustering + SVR

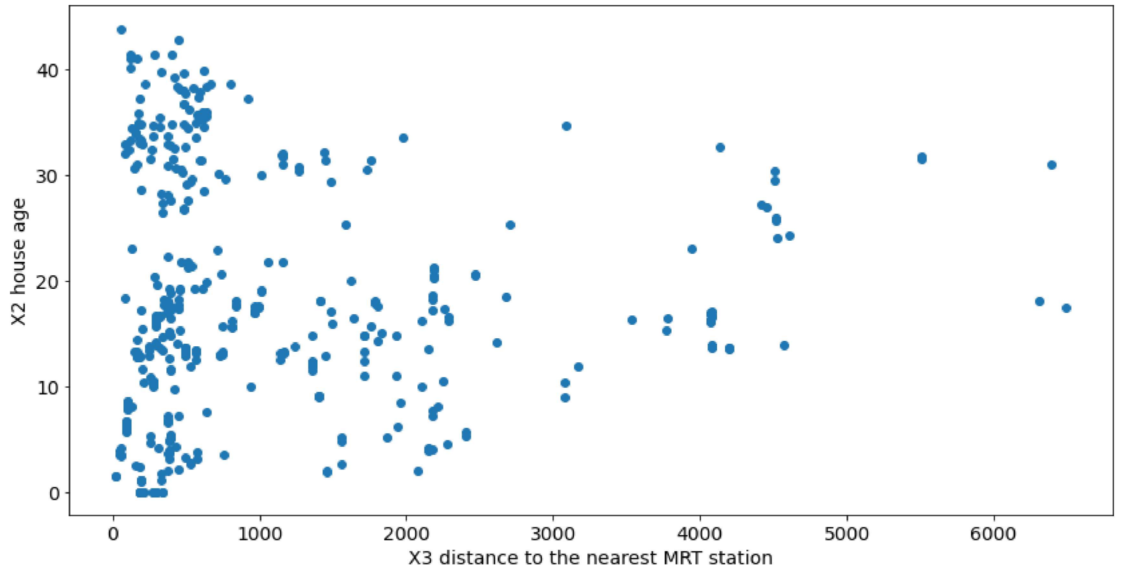
In [7]:
```
# Read data into a dataframe
df = pd.read_csv('Real estate.csv', encoding='utf-8')
df.head()
```

Out[7]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

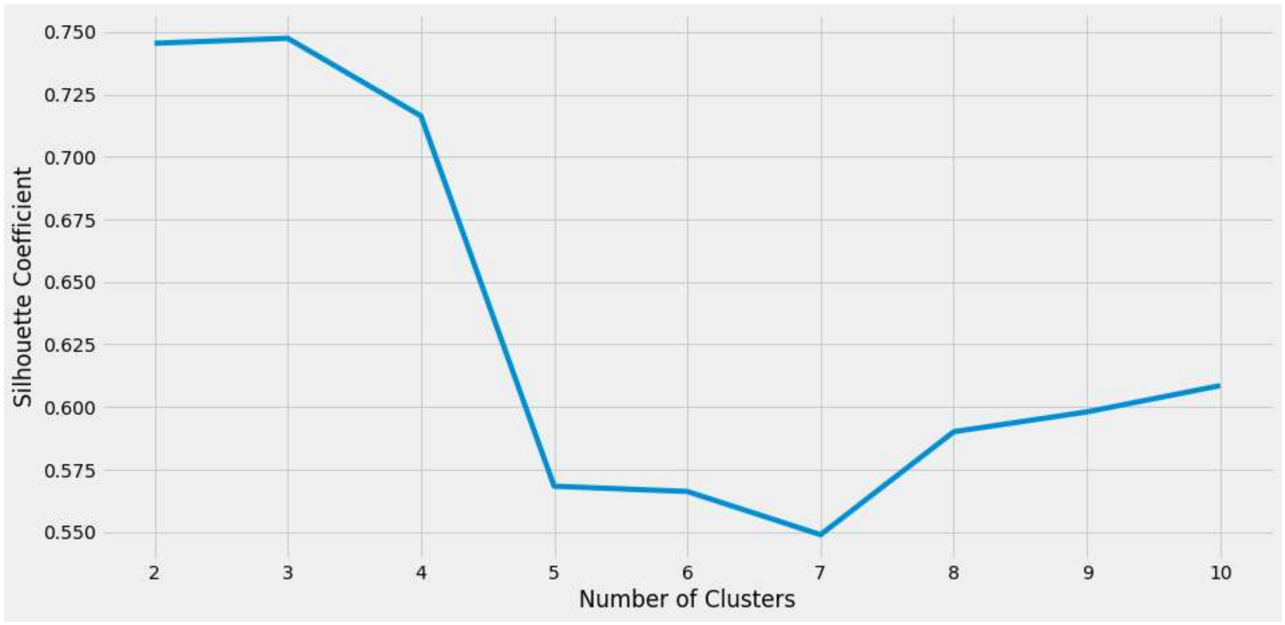
In [8]:
```
# Defining Dependent and Independant variable
X = np.array(df[['X3 distance to the nearest MRT station','X2 house age']])
Y = df['Y house price of unit area'].values
```

In [9]:
```
# Plotting the Clusters using matplotlib
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [14, 7]
plt.rc('font', size=14)

plt.scatter(df['X3 distance to the nearest MRT station'],df['X2 house age'])
plt.xlabel("X3 distance to the nearest MRT station")
plt.ylabel("X2 house age")
plt.show()
```



In [10]:
```
# Split and train data KMeans
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=42)

silhouette_coefficients = []

kmeans_kwargs= {
    "init":"random",
    "n_init":10,
    "max_iter":300,
    "random_state":42
}

for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(X_train)
    score = silhouette_score(X_train, kmeans.labels_)
    silhouette_coefficients.append(score)
```

In [11]:
```
# Plotting graph to choose the best number of clusters
# with the most Silhouette Coefficient score

plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```

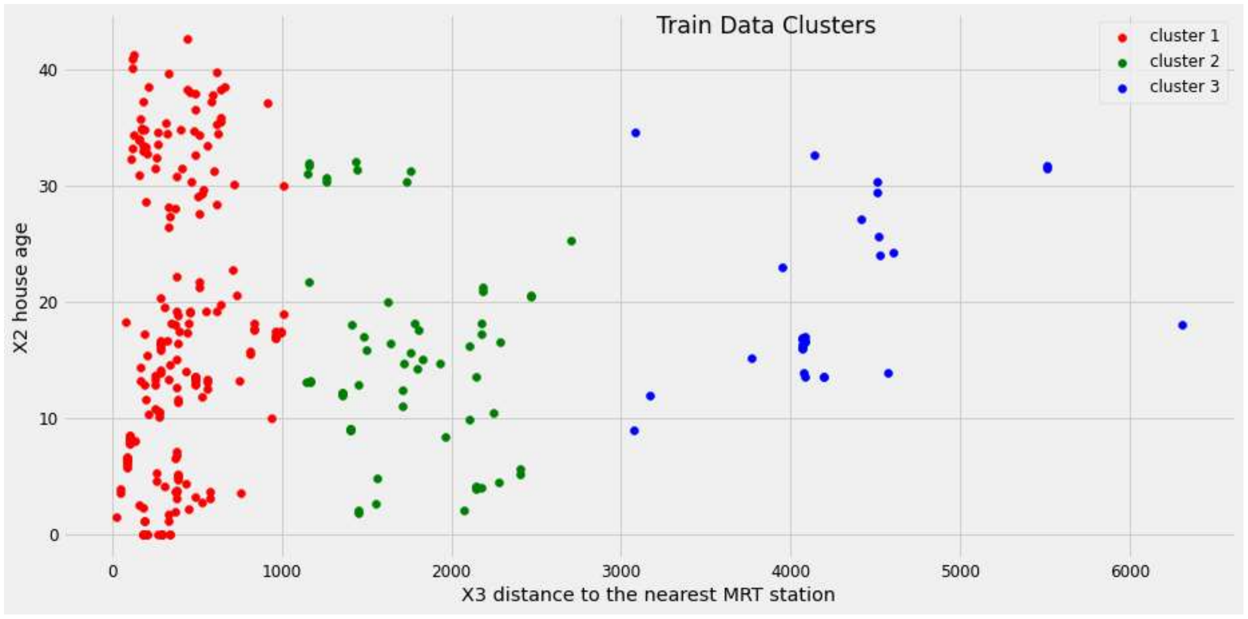


We can arrive at the conclusion that with 3 clusters, we can achieve better clustering. Let's go ahead and cluster the training set data.

In [12]:
```
# Instantiate the model: KMeans from sklearn
kmeans = KMeans(
    init="random",
    n_clusters=3,
    n_init=10,
    max_iter=300,
    random_state=42
)

# Fit to the training data
kmeans.fit(X_train)

train_df = pd.DataFrame(X_train,columns=['X3 distance to the nearest MRT station','X2 house age'])
```

In [13]:
```
# Generate out clusters
train_cluster = kmeans.predict(X_train)

# Add the target and predicted clusters to our training DataFrame
train_df.insert(2,'Y house price of unit area',Y_train)
train_df.insert(3,'cluster',train_cluster)

n_clusters=3
train_clusters_df = []
for i in range(n_clusters):
    train_clusters_df.append(train_df[train_df['cluster']==i])

colors = ['red','green','blue']
plt.rcParams['figure.figsize'] = [14, 7]
plt.rc('font', size=12)
```

In [14]:
```
# Plot X_train again with features labeled by cluster
for i in range(n_clusters):
  subset = []
  for count,row in enumerate(X_train):
      if(train_cluster[count]==i):
        subset.append(row)

  x = [row[0] for row in subset]
  y = [row[1] for row in subset]
  plt.scatter(x,y,c=colors[i],label="cluster "+ str(i+1))
plt.title("Train Data Clusters", x=0.6, y=0.95)
plt.xlabel("X3 distance to the nearest MRT station")
plt.ylabel("X2 house age")
plt.legend()
plt.show()
```



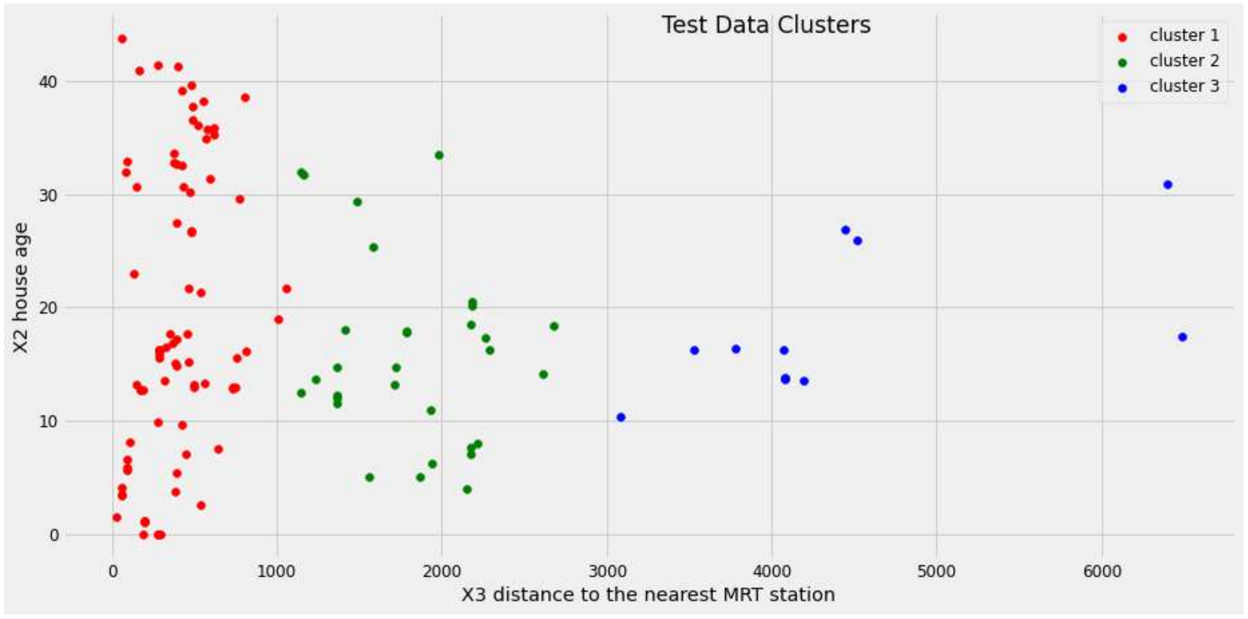
Let us define a function that will predict House Price (Y) by predicting the cluster first, and then the value with the corresponding SVR.

Building SVR for clusters

In [15]:
```
n_clusters=3
cluster_svr = []
model = SVR(kernel='rbf', C=1000, epsilon=1)

for i in range(n_clusters):
    cluster_X = np.array((train_clusters_df[i])[['X3 distance to the nearest MRT station','X2 house age']])
    cluster_Y = (train_clusters_df[i])['Y house price of unit area'].values
    cluster_svr.append(model.fit(cluster_X, cluster_Y))
```

In [16]:
```
def regression_function(arr, kmeans, cluster_svr):
    result = []
    clusters_pred = kmeans.predict(arr)
    for i,data in enumerate(arr):
        result.append(((cluster_svr[clusters_pred[i]]).predict([data]))[0])
    return result,clusters_pred
```

In [17]:
```
# calculating Y value and cluster
Y_svr_k_means_pred, Y_clusters = regression_function(X_test, kmeans, cluster_svr)

colors = ['red','green','blue']
plt.rcParams['figure.figsize'] = [14, 7]
plt.rc('font', size=12)

n_clusters=3

# Apply our model to clustering the remaining test set for validation

for i in range(n_clusters):
  subset = []
  for count,row in enumerate(X_test):
      if(Y_clusters[count]==i):
        subset.append(row)

  x = [row[0] for row in subset]
  y = [row[1] for row in subset]
  plt.scatter(x,y,c=colors[i],label="cluster "+ str(i+1))
plt.title("Test Data Clusters", x=0.6, y=0.95)
plt.xlabel("X3 distance to the nearest MRT station")
plt.ylabel("X2 house age")
plt.legend()
plt.show()
```



In [18]:
```
result_df = pd.DataFrame(X_test,columns=['X3 distance to the nearest MRT station','X2 house age'])
result_df['Y true value'] =  Y_test
result_df['Y SVR + K means'] = Y_svr_k_means_pred
result_df['cluster'] = Y_clusters
result_df.head()
```

Out[18]:

	X3 distance to the nearest MRT station	X2 house age	Y true value	Y SVR + K means	cluster
0	193.5845	1.1	45.1	70.943672	0
1	492.2313	13.2	42.3	74.662003	0
2	274.0144	0.0	52.2	72.037664	0
3	170.1289	12.7	37.3	70.631881	0
4	2185.1280	20.2	22.8	64.385493	1