In [33]:
```
import pandas as pd
import numpy as np
```

In [34]:
```
df = pd.read_csv('Caravan.csv')
df = df.set_index('Unnamed: 0')
df.head(5)
```

Out[34]:

Unnamed: 0	MOSTYPE	MAANTHUI	MGEMOMV	MGEMLEEF	MOSHOOFD	MGODRK	MGODPR	MG
1	33	1	3	2	8	0	5	
2	37	1	2	2	8	1	4	
3	37	1	2	2	8	0	4	
4	9	1	3	3	3	2	3	
5	40	1	4	2	10	1	4	

5 rows × 86 columns

Q.no. 1. a

In [35]:
```
print("The dimension of the dataset is", df.shape)
```

```
The dimension of the dataset is (5822, 86)
```

Q.no. 1. b

In [36]:
```
print("The number of predictors that measure demographics characters are", 43)
```

```
The number of predictors that measure demographics characters are 43
```

Q.no. 1. c

In [37]:
```
per_yes = round((df[df['Purchase'] == 'Yes'].shape[0] / df.shape[0]) * 100, 2)
print("The percentage of people who purchased caravan insurance is", per_yes, "%"
```

```
The percentage of people who purchased caravan insurance is 5.98 %
```

Q.no. 2.

In [38]:
```
from sklearn import preprocessing
df.loc[:, df.columns!='Purchase'] = preprocessing.scale(df.loc[:, df.columns!='Pu
df.describe()
```

Out[38]:

	MOSTYPE	MAANTHUI	MGEMOMV	MGEMLEEF	MOSHOOFD	MGODRK	
count	5.822000e+03	5.822000e+03	5.822000e+03	5.822000e+03	5.822000e+03	5.822000e+03	5.
mean	-1.015639e-16	-5.891695e-16	5.059123e-17	2.968433e-16	3.783378e-17	8.966453e-17	2
std	1.000086e+00	1.000086e+00	1.000086e+00	1.000086e+00	1.000086e+00	1.000086e+00	1.
min	-1.810219e+00	-2.725800e-01	-2.125697e+00	-2.444683e+00	-1.671134e+00	-6.943106e-01	-2.
25%	-1.109590e+00	-2.725800e-01	-8.595001e-01	-1.216964e+00	-9.709796e-01	-6.943106e-01	-3
50%	4.473633e-01	-2.725800e-01	4.066966e-01	1.075466e-02	4.293284e-01	-6.943106e-01	2
75%	8.366015e-01	-2.725800e-01	4.066966e-01	1.075466e-02	7.794054e-01	3.025516e-01	8
max	1.303687e+00	2.190544e+01	2.939090e+00	3.693911e+00	1.479559e+00	8.277450e+00	2.

8 rows × 85 columns

All the variables in the datset now have a mean of 0 and standard deviation of 1 and hence are standardized.

Q.no. 3.

In [39]:
```
X, y = df.drop(['Purchase'], axis = 1), df['Purchase']
X_train, X_test, y_train, y_test = X[1000:], X[:1000], y[1000:], y[:1000]

y_train = (y_train == 'Yes')
y_test = (y_test == 'Yes')
```

Q.no. 3.a

In [40]:
```
print("The number of observations in train set is", X_train.shape[0])
print("The number of observations in test set is", X_test.shape[0])
```

```
The number of observations in train set is 4822
The number of observations in test set is 1000
```

Q.no. 3.b

In [41]:
```
print("The number of people purchasing the insurance in train set is", np.count_
print("The number of people purchasing the insurance in test set is", np.count_nc
```

```
The number of people purchasing the insurance in train set is 289
The number of people purchasing the insurance in test set is 59
```

Q.no. 4.a

In [42]:
```
from sklearn.neighbors import KNeighborsClassifier

neigh1 = KNeighborsClassifier(n_neighbors=1)
neigh1.fit(X_train, y_train)
y_pred1 = neigh1.predict(X_test)

neigh3 = KNeighborsClassifier(n_neighbors=3)
neigh3.fit(X_train, y_train)
y_pred3 = neigh3.predict(X_test)

neigh5 = KNeighborsClassifier(n_neighbors=5)
neigh5.fit(X_train, y_train)
y_pred5 = neigh5.predict(X_test)
```

In [43]:
```
from sklearn.metrics import precision_score, recall_score
print("K value = 1")
print("Precision score", precision_score(y_test, y_pred1))
print("Recall score", recall_score(y_test, y_pred1))
print("=" * 40)
print("K value = 3")
print("Precision score", precision_score(y_test, y_pred3))
print("Recall score", recall_score(y_test, y_pred3))
print("=" * 40)
print("K value = 5")
print("Precision score", precision_score(y_test, y_pred5))
print("Recall score", recall_score(y_test, y_pred5))
```

```
K value = 1
Precision score 0.11688311688311688
Recall score 0.15254237288135594
========================================
K value = 3
Precision score 0.2
Recall score 0.0847457627118644
========================================
K value = 5
Precision score 0.26666666666666666
Recall score 0.06779661016949153
```

As we increase the number of K, the precision score of the model increases which means the accuracy of the positive predictions (the number of positively predicted classes actually turning out to be positive) increase as the number of K is increased.

Q.no. 4.b

In [44]:
```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
sgd_clf.fit(X_train, y_train)

sgd_pred = sgd_clf.predict(X_test)
print("Precision score", precision_score(y_test, sgd_pred))
print("Recall score", recall_score(y_test, sgd_pred))
```

```
Precision score 0.3125
Recall score 0.0847457627118644
```

Q.no. 4.c

The precision score of SGD classifier is much higher than any of the three models(K=1, 3, and 5) of the KNN. In our case, we want any customer to be predicted as potential buyer only when the chances are really high as the insurance company doesn't want to spend much time and money on those who are not likely to buy. By doing so, they might miss few potential buyers but here time and other forms of investments might be crucial aspect. In other words, we want the precision to be high with less regard to the recall value. Hence, SGD classifier is said to capture real patterns in the Caravan dataset.