NPTEL PYTHON FOR DATA SCIENCE - ASSIGNMENT 4 - SOLUTION

Given Data: Credit Worthiness data containing 1000 observations of income details of individuals comprising 21 attributes along the columns (Cbal, Cdur, Chist, Cpur, Camt, Sbal, Edur, InRate, MSG, Oparties, Rdur, Prop, age, inPlans, Htype, NumCred, JobType, Ndepend, telephone, foreign, creditScore)

Problem statement: By observing the features of the dataset, the problem statement can be defined as a binary classification problem of classifying any individual into an appropriate category of creditScore such as Good or Bad.

1. **To identify invalid values in categorical columns of the dataframe, which of the following will be very helpful and less time consuming in python?**

   a) View the data manually and find the invalid values

   b) By observing unique values of every categorical column of the data using unique() method

   c) Only by viewing info of the dataframe using info() method

   d) By estimating the statistics of the data using describe() method

Ans: b

2. **Which of the following columns in the CreditWorthiness dataset contains invalid value in it and find the invalid values?**

   I. JobType

   II. Htype

   III. foreign

   IV. None of the above

a) I & III and the invalid value is ‘?’

b) IV since there are no invalid values

c) Both I and II and the invalid value is ‘?’

d) Both II and III and the invalid value is ‘?’

Ans: b

In [1]:

```
import pandas as pd
import numpy as np
# Reading the data set
data=pd.read_excel('CreditWorthiness.xlsx','Data')#,na_values=['?'])
```

In [2]:

```
categotical_data = data.select_dtypes(include=['object']).copy()

print(categotical_data.columns)

#categotical_data = categotical_data.drop(['customerID','tenure'],axis = 1)

frequencies      = categotical_data.apply(lambda x: x.value_counts()).T.stack()

print(frequencies)
```

```
3.0
             Rs. < 1000                                          60
3.0
             Rs. >= 10,000                                        4
8.0
             no savings account                                  18
3.0
Edur         1 to 4 years                                        33
9.0
             4 to 7 years                                        17
4.0
             less than 1 year                                    17
2.0
             more than 7 years                                   25
3.0
             not employed                                         6
2.0
MSG          divorced or separated male                           5
0.0
```

3. **For the given CreditWorthiness data choose the appropriate problem and technique from the following options. The options should be chosen from machine learning perspective by considering the target variable as creditScore (Y) and other variables as input features (X)**

   I. Binary classification problem

   II. Supervised machine learning technique

   III. Unsupervised machine learning technique

   IV. Linear regression problem

Solution: c) I and II

This is a binary classification problem of classifying any individual into an appropriate category of creditScore

such as Good or Bad. And the classification comes under supervised machine learning technique

4. **Which of the following machine learning techniques would be applicable to solve the problem given in the problem statement?**

   Solution: a) KNN b) Random Forest and d) Logistic Regression

5. **The most linearly correlated feature set in the given dataset is?**

   ****Solution: c) Cdur and Camt**

In [3]:

```
data.corr()
```

Out[3]:

	Cdur	Camt	InRate	age	NumCred	Ndepend
Cdur	1.000000	0.624984	0.074749	-0.036136	-0.011284	-0.023834
Camt	0.624984	1.000000	-0.271316	0.032716	0.020795	0.017142
InRate	0.074749	-0.271316	1.000000	0.058266	0.021669	-0.071207
age	-0.036136	0.032716	0.058266	1.000000	0.149254	0.118201
NumCred	-0.011284	0.020795	0.021669	0.149254	1.000000	0.109667
Ndepend	-0.023834	0.017142	-0.071207	0.118201	0.109667	1.000000

6. **What is the average age of the borrower whose balance in checking account is less than zero?**

   Solution: b) 35

In [ ]:

```
data[data["Cbal"] == ' Rs. < 0']["age"].describe()
```

7. **What is the average loan amount claimed by borrower for the purpose of education?**

   Solution: a) 31684

In [4]:

```
nominal_features=data.select_dtypes(include=['object']).columns
```

In [9]:

```
# import library for plotting
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline
for col in nominal_features:
    bp=data.boxplot('Camt',col,rot = 30,figsize=(15,6),return_type='both')
    for name, group in data.groupby(col):
        print('\033[1m'+str(name)+'\033[0m')
        print(group.Camt.describe())
    plt.show()
```

```
50%       31490.000000
75%       52810.000000
max      159330.000000
Name: Camt, dtype: float64
domestic needs
count       12.000000
mean     14860.000000
std      10125.107407
min       3310.000000
25%      11192.500000
50%      12370.000000
75%      13475.000000
max      39780.000000
Name: Camt, dtype: float64
education
count        50.000000
mean      31684.000000
std       30179.420005
min        3800.000000
```

8. **How many nearest neighbors can be considered for the most accurate prediction of the creditscore with train:test split of 75%:25% over encoded (appropriate encoding) dataset (without standardising) with random_state=1 and metric=euclidean? Note: k is tested with values ranging from 1 to 25**

   Solution: d) 23

In [10]:

```
# -*- coding: utf-8 -*-
"""
Importing necessary libraries
"""
import pandas as pd
import numpy as np

# import library for plotting
import matplotlib.pyplot as plt
import seaborn as sns2
sns.set()
%matplotlib inline
```

In [11]:

```
# Reading the data set
data=pd.read_excel('CreditWorthiness.xlsx','Data')#,na_values=['?'])
```

KNN Classification

In [12]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


x = pd.get_dummies(data.drop(columns=['creditScore']),drop_first=True)
y = data['creditScore']

train_x, test_x, train_y, test_y = train_test_split( x, y, test_size=0.25, random_state=1)
```

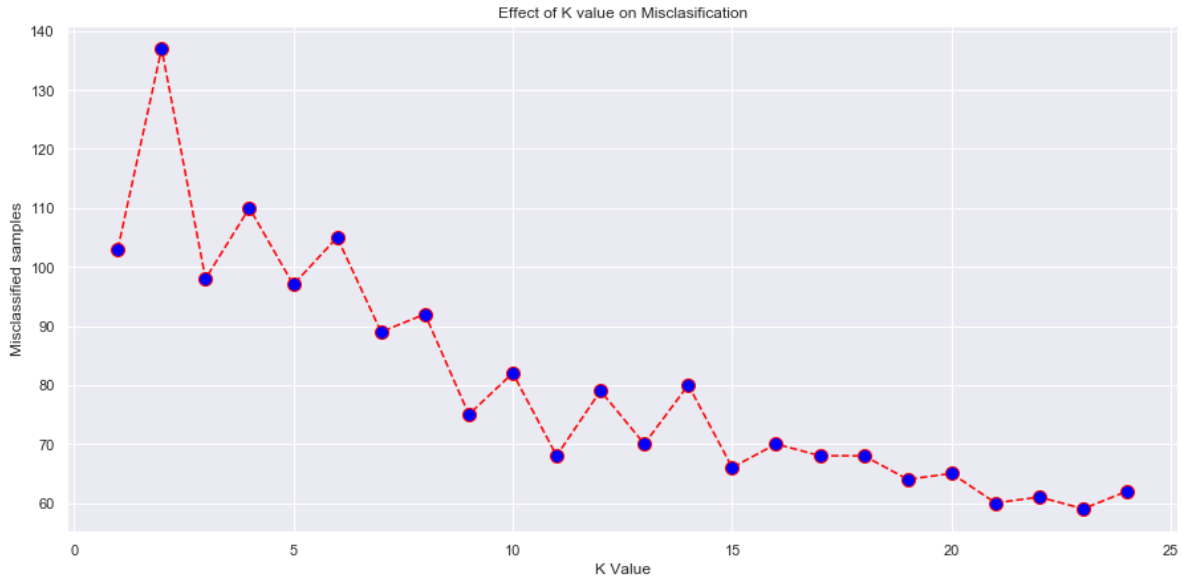
In [13]:

```
Misclassified_sample = []
accuracy_scores=[]
# Calculating error for K values between 1 and 40
for i in range(1, 25):
    knn = KNeighborsClassifier(n_neighbors=i,metric='euclidean')
    knn.fit(train_x, train_y)
    pred_i = knn.predict(test_x)
    Misclassified_sample.append((test_y != pred_i).sum())
    accuracy_scores.append(accuracy_score(test_y, pred_i))
print(accuracy_scores)
```

```
[0.588, 0.452, 0.608, 0.56, 0.612, 0.58, 0.644, 0.632, 0.7, 0.672, 0.728, 0.
684, 0.72, 0.68, 0.736, 0.72, 0.728, 0.728, 0.744, 0.74, 0.76, 0.756, 0.764,
0.752]
```

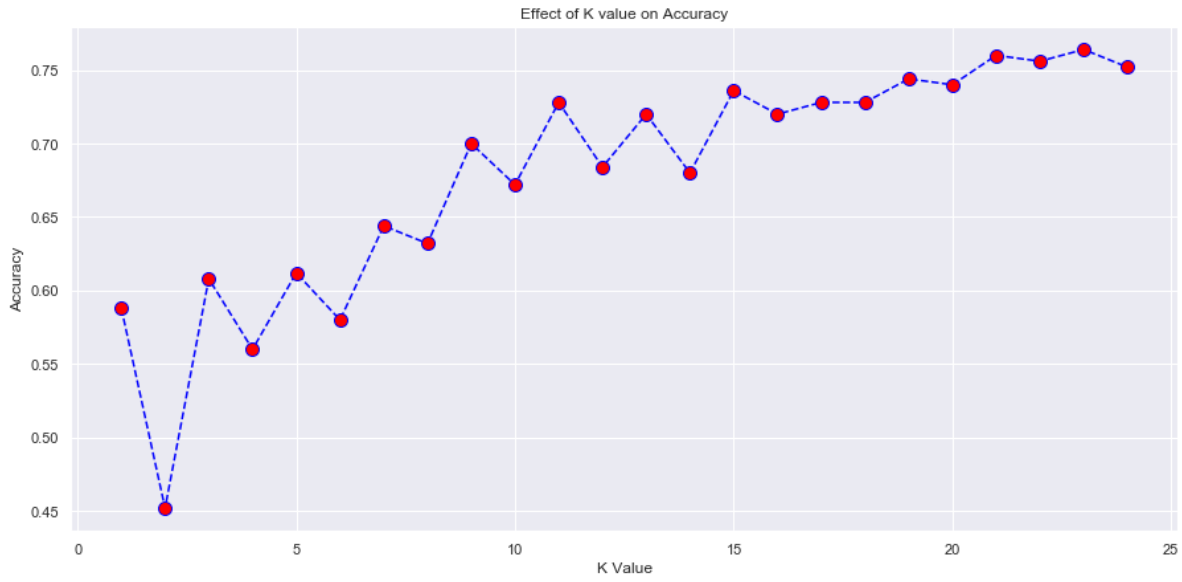
In [14]:

```
# Plotting the effect of K
plt.figure(figsize = (15,7))
plt.plot(range(1,25,1),Misclassified_sample,
         color='red',linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Effect of K value on Misclasification')
plt.xlabel('K Value')
plt.ylabel('Misclassified samples')
plt.show()
```



In [15]:

```
# Plotting the effect of K
plt.figure(figsize = (15,7))
plt.plot(range(1,25,1),accuracy_scores,
         color='blue',linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Effect of K value on Accuracy')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.show()
```



9. **What is the accuracy (in percentage) of the model built in Q 8 for test data?**

   Solution: a) 76.4

In [16]:

```
KNN_classifier = KNeighborsClassifier(n_neighbors = 23,metric = 'euclidean')
KNN_classifier.fit(train_x, train_y)

prediction = KNN_classifier.predict(test_x)
```

In [17]:

```
confusion_matrix1 = confusion_matrix(test_y, prediction)
print(confusion_matrix1)
```

```
[[ 11  51]
 [  8 180]]
```

In [18]:

```
accuracy_score1 = accuracy_score(test_y, prediction)
print(accuracy_score1)
```

```
0.764
```

10. **By standardising (using StatndardScaler of sklearn) the numerical features in the data using the same model confidguration as in Q8, the test data accuracy gets improved by approximately? Ans: 1.6%**

    Solution:

In [19]:

```
x = pd.get_dummies(data.drop(columns=['creditScore']),drop_first=True)
y = data['creditScore']
train_x, test_x, train_y, test_y = train_test_split( x, y, test_size=0.25, random_state=1)
```

In [20]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
col_names = ['age', 'Camt','Cdur','InRate','NumCred','Ndepend']
features = train_x[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
train_x.loc[:,col_names]=features
features = test_x[col_names]
features = scaler.transform(features.values)
test_x.loc[:,col_names]=features
#print('actual=',train_x.age.shape,' ,  ','standard=',abc.shape)

#train_x.Camt = pd.Series(sc.fit_transform(train_x.Camt))
#test_x.age = pd.Series(sc.transform(test_x.age))
#test_x.Camt = pd.Series(sc.transform(test_x.Camt))
```

```
c:\users\gitaa\spyder\venv\lib\site-packages\pandas\core\indexing.py:494: Se
ttingWithCopyWarning:
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: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  self.obj[item] = s
c:\users\gitaa\spyder\venv\lib\site-packages\pandas\core\indexing.py:494: Se
ttingWithCopyWarning:
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: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  self.obj[item] = s
```

In [21]:

```
train_x.head()
```

Out[21]:

	Cdur	Camt	InRate	age	NumCred	Ndepend	Cbal_0 <= Rs. < 2000	Cbal_Rs. >=2000	Cbal_r checkir accou
298	-0.741155	-0.905203	0.902335	-0.675430	-0.711202	-0.434269	0	0	
160	-0.741155	-0.960829	0.902335	-0.852801	-0.711202	-0.434269	1	0	
268	1.266186	0.456709	0.902335	-0.409373	-0.711202	-0.434269	0	1	
658	0.262516	-0.031399	0.011873	-0.409373	1.015017	-0.434269	0	1	
996	-0.741155	1.191633	-0.878590	1.453026	-0.711202	-0.434269	1	0	

5 rows × 49 columns

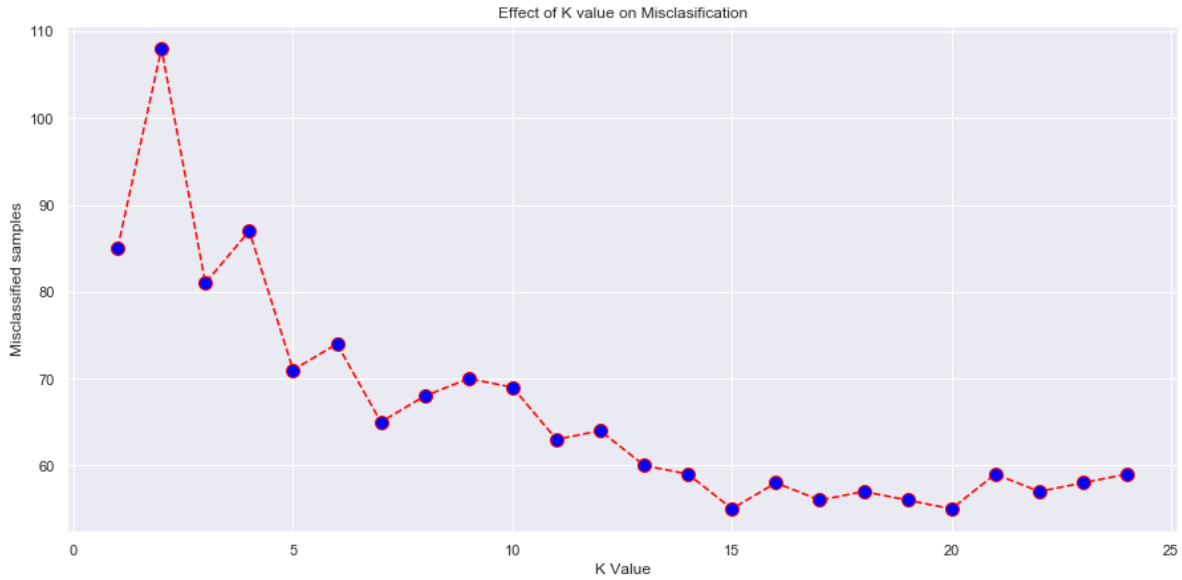
In [22]:

```
Misclassified_sample = []
accuracy_scores=[]
# Calculating error for K values between 1 and 40
for i in range(1, 25):
    knn = KNeighborsClassifier(n_neighbors=i) # by default, Minkowski with p = 2 --> same d
    knn.fit(train_x, train_y)
    pred_i = knn.predict(test_x)
    Misclassified_sample.append((test_y != pred_i).sum())
    accuracy_scores.append(accuracy_score(test_y, pred_i))
print(accuracy_scores)
```

```
[0.66, 0.568, 0.676, 0.652, 0.716, 0.704, 0.74, 0.728, 0.72, 0.724, 0.748,
0.744, 0.76, 0.764, 0.78, 0.768, 0.776, 0.772, 0.776, 0.78, 0.764, 0.772, 0.
768, 0.764]
```

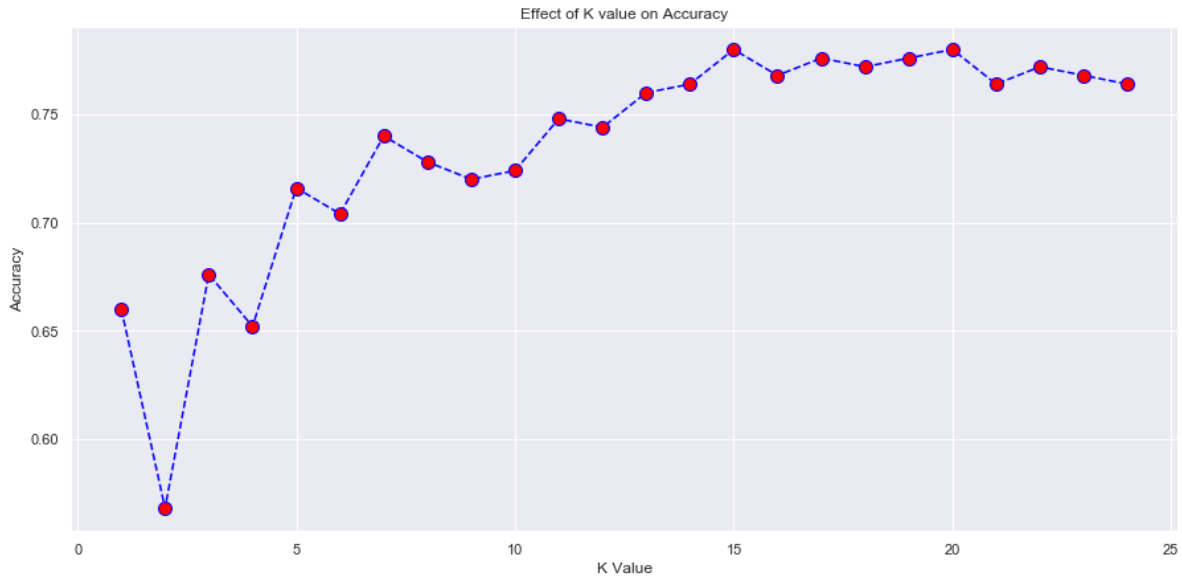
In [23]:

```
# Plotting the effect of K
plt.figure(figsize = (15,7))
plt.plot(range(1,25,1),Misclassified_sample,
         color='red',linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Effect of K value on Misclasification')
plt.xlabel('K Value')
plt.ylabel('Misclassified samples')
plt.show()
```



In [24]:

```
# Plotting the effect of K
plt.figure(figsize = (15,7))
plt.plot(range(1,25,1),accuracy_scores,
         color='blue',linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Effect of K value on Accuracy')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.show()
```



In [25]:

```
KNN_classifier = KNeighborsClassifier(n_neighbors = 15,metric = 'euclidean')
KNN_classifier.fit(train_x, train_y)

prediction = KNN_classifier.predict(test_x)
```

In [26]:

```
confusion_matrix1 = confusion_matrix(test_y, prediction)
tn, fp, fn, tp = confusion_matrix(test_y, prediction).ravel()
print(confusion_matrix1)
print('tp:',tp,'tn:',tn,'fp:',fp,'fn:',fn)
```

```
[[ 17  45]
 [ 10 178]]
tp: 178 tn: 17 fp: 45 fn: 10
```

In [27]:

```
accuracy_score1 = accuracy_score(test_y, prediction)
print(accuracy_score1)
```

```
0.78
```

11. **By standardising (using StatndardScaler of sklearn) the numerical features in the data using the same model confidguration as in Q8, the optimum value for 'k' in knn model is found to be?**

Ans: b) 15

12. **Which type of error is large with the above model for the test dataset?**

    Solution: b) Type-I error

13. **Choose the correct answer(s) based on the following confusion matrix obtained using sklearn.metrics.confusion_matrix method***

    [[33 29]

    [27 161]]

    Solution: b) and d)

False Positives = 29, False Negatives = 27, True Positives = 161, True Negatives=33

14. **What is the test data accuracy of logistic regression model over the same standardised and encoded dataset in percentage approximately?**

    Solution: a) 78

Logistic Regression over encoded, resampled (upsampled) dataset used for latest knn modeling

In [ ]:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(max_iter=10000)
logreg.fit(train_x,train_y)
prediction = logreg.predict(test_x)
```

#prediction=le.transform(prediction) test_y=le.transform(test_y)

In [ ]:

```
confusion_matrix1 = confusion_matrix(test_y, prediction)
tn, fp, fn, tp = confusion_matrix1.ravel()
print(confusion_matrix1)
print('tp:',tp,'tn:',tn,'fp:',fp,'fn:',fn)
```

In [ ]:

```
precision = tp/(tp+fp)
recall = tp/(tp+fn)
f1_score = 2/(1/precision + 1/recall)
print(precision, recall, f1_score)
```

In [ ]:

```
accuracy_score1 = accuracy_score(test_y, prediction)
print(accuracy_score1)
```

15. **State whether the following statement is True/False**

    Positive predictive value and precision are synonyms to each other

    Solution: a) True