In [1]:
```
#Import dataset da libreria scikit-learn
from sklearn import datasets
import pandas as pd

#Caricamento dataset
data = pd.read_csv("dataset2.csv ", header = 0)
#data.head()
```

In [14]:
```
# Import train_test_split function
from sklearn.model_selection import train_test_split

#Dal mio dataset vado a separare le feature e le classi
y=data["fornitura(20-50-100-150-200-300-500)"]
#converte le stringhe in set di boolean (One-hot encode)
X = pd.get_dummies(data[["tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)","stagione","zona supermercato(Periferia,Residenziale)","festività(Feriale,Lavorativo)","scadenza(Breve,Media,Lunga)","dimensione confezione(Piccola,Media,Grande)","costo(prezzo)","spedizione(prezzo)"]])
#print(X)

# Split dataset nel training set e test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training e 30% test
#print(X_test,y_test)
```

In [15]:
```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Creazione classificatore con alberi dispari
clf=RandomForestClassifier(n_estimators=99)

#bootstrap=vengono utilizzati sottinsieme di variabili, random_state=sottinsiemi di variabili casuali
clf=RandomForestClassifier(n_estimators=99,bootstrap=True,random_state=0)


#Addestramento
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
#print(X_test)

#res=clf.predict([[15,15,1,0,0,0,0,1,1,0,0,0,1,1,0,0,1,0,0,1]])
#print(res)
```

In [16]:
```
#Import scikit-learn metrics module per il calcolo dell'accuracy
from sklearn import metrics
# Model Accuracy, quanto spesso il classificatore è corretto?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#print(y_test, y_pred)
```

```
Accuracy: 0.8245614035087719
```

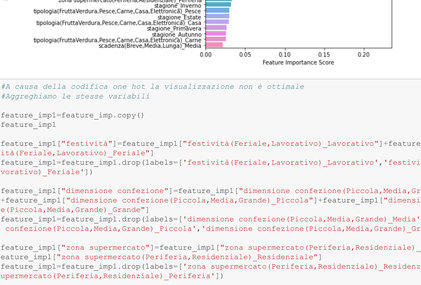
In [17]:
```
#Series rappresenta dati 1D
#Mostriamo l'importanza delle feature ordinate in ordine decrescente
feature_imp = pd.Series(clf.feature_importances_,index=X.columns).sort_values(ascending=False)
feature_imp
```

Out[17]:
```
costo(prezzo)                                                                  0.223911
spedizione(prezzo)                                                             0.115466
dimensione confezione(Piccola,Media,Grande)_Grande                             0.050331
dimensione confezione(Piccola,Media,Grande)_Piccola                            0.049490
dimensione confezione(Piccola,Media,Grande)_Media                              0.046266
tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_FruttaVerdura            0.044518
festività(Feriale,Lavorativo)_Lavorativo                                       0.040242
tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Elettronica              0.037111
festività(Feriale,Lavorativo)_Feriale                                          0.036560
scadenza(Breve,Media,Lunga)_Lunga                                              0.034247
scadenza(Breve,Media,Lunga)_Breve                                              0.033796
zona supermercato(Periferia,Residenziale)_Residenziale                         0.033549
zona supermercato(Periferia,Residenziale)_Periferia                            0.033531
stagione_Inverno                                                               0.031828
tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Pesce                    0.030381
stagione_Estate                                                                0.029753
tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Casa                     0.029467
stagione_Primavera                                                             0.026340
stagione_Autunno                                                               0.025994
tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Carne                    0.025633
scadenza(Breve,Media,Lunga)_Media                                              0.021584
dtype: float64
```

In [18]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Creiamo un bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)

# Aggiungiamo le etichette
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
#plt.legend()
plt.show()
```



In [19]:
```
#A causa della codifica one hot la visualizzazione non è ottimale
#Aggreghiamo le stesse variabili

feature_imp1=feature_imp.copy()
feature_imp1

feature_imp1["festività"]=feature_imp1["festività(Feriale,Lavorativo)_Lavorativo"]+feature_imp1["festività(Feriale,Lavorativo)_Feriale"]
feature_imp1=feature_imp1.drop(labels=['festività(Feriale,Lavorativo)_Lavorativo','festività(Feriale,Lavorativo)_Feriale'])

feature_imp1["dimensione confezione"]=feature_imp1["dimensione confezione(Piccola,Media,Grande)_Media"]+feature_imp1["dimensione confezione(Piccola,Media,Grande)_Piccola"]+feature_imp1["dimensione confezione(Piccola,Media,Grande)_Grande"]
feature_imp1=feature_imp1.drop(labels=['dimensione confezione(Piccola,Media,Grande)_Media','dimensione confezione(Piccola,Media,Grande)_Piccola','dimensione confezione(Piccola,Media,Grande)_Grande'])

feature_imp1["zona supermercato"]=feature_imp1["zona supermercato(Periferia,Residenziale)_Periferia"]+feature_imp1["zona supermercato(Periferia,Residenziale)_Residenziale"]
feature_imp1=feature_imp1.drop(labels=['zona supermercato(Periferia,Residenziale)_Residenziale','zona supermercato(Periferia,Residenziale)_Periferia'])

feature_imp1["scadenza"]=feature_imp1["scadenza(Breve,Media,Lunga)_Media"]+feature_imp1["scadenza(Breve,Media,Lunga)_Lunga"]+feature_imp1["scadenza(Breve,Media,Lunga)_Breve"]
feature_imp1=feature_imp1.drop(labels=['scadenza(Breve,Media,Lunga)_Media','scadenza(Breve,Media,Lunga)_Lunga','scadenza(Breve,Media,Lunga)_Breve'])

feature_imp1["stagione"]=feature_imp1["stagione_Primavera"]+feature_imp1["stagione_Estate"]+feature_imp1["stagione_Autunno"]+feature_imp1["stagione_Inverno"]
feature_imp1=feature_imp1.drop(labels=['stagione_Inverno','stagione_Estate','stagione_Primavera','stagione_Autunno'])

feature_imp1["tipologia"]=feature_imp1["tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Elettronica"]+feature_imp1["tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Carne"]+feature_imp1["tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Pesce"]+feature_imp1["tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_FruttaVerdura"]+feature_imp1["tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Casa"]
feature_imp1=feature_imp1.drop(labels=['tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Elettronica','tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Carne','tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Pesce','tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_FruttaVerdura','tipologia(FruttaVerdura,Pesce,Carne,Casa,Elettronica)_Casa'])


feature_imp1.sort_values(ascending=False)
```

Out[19]:
```
costo(prezzo)             0.223911
tipologia                 0.167111
dimensione confezione     0.146087
spedizione(prezzo)        0.115466
stagione                  0.113915
scadenza                  0.089628
festività                 0.076802
zona supermercato         0.067080
dtype: float64
```

In [20]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Ricreiamo il bar plot
sns.barplot(x=feature_imp1, y=feature_imp1.index)


plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
#plt.legend()
plt.show()
```

In [21]:
```
# Tools per la visualizzazione
from sklearn.tree import export_graphviz
import pydot

# Prendiamo un albero dalla foresta
tree = clf.estimators_[5]

feature_list=list(X.columns)
# Esportiamo l'immagine
export_graphviz(tree, out_file = 'tree.dot', feature_names = feature_list, rounded = True, precision = 1)

# Usiamo un file dot per salvare l'immagine
(graph, ) = pydot.graph_from_dot_file('tree.dot')

# Trasformiamo in png
graph.write_png('tree.png')
```

In [22]:
```
from IPython.display import Image
Image(filename='tree.png')
```

Out[22]:

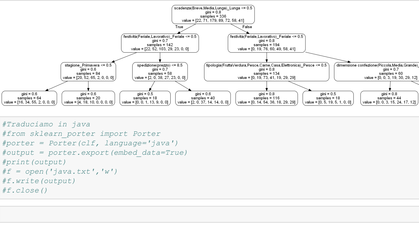
In [23]:
```
# Limitiamo la profondità a 3 livelli
rf_small = RandomForestClassifier(n_estimators=10, max_depth = 3)
rf_small.fit(X_train,y_train)

# Estraiamo l'albero ridotto
tree_small = rf_small.estimators_[5]

# Salviamo come png
export_graphviz(tree_small, out_file = 'small_tree.dot', feature_names = feature_list, rounded = True, precision = 1)
(graph, ) = pydot.graph_from_dot_file('small_tree.dot')
graph.write_png('small_tree.png');

# Mostriamo
Image(filename='small_tree.png')
```

Out[23]:



In [18]:
```
#Traduciamo in java
#from sklearn_porter import Porter
#porter = Porter(clf, language='java')
#output = porter.export(embed_data=True)
#print(output)
#f = open('java.txt','w')
#f.write(output)
#f.close()
```

In [ ]:

In [ ]: