In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split, cross_val_score
```

```
Bad key "text.kerning_factor" on line 4 in
C:\Users\cyine\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test_pa
tch.mplstyle.
You probably need to get an updated matplotlibrc file from
http://github.com/matplotlib/matplotlib/blob/master/matplotlibrc.template
or from the matplotlib source distribution
```

For data preprocessing, missing values were dropped and only personal types were included.

In [2]:
```
df = pd.read_csv("C:\\Users\\cyine\\Desktop\\processed_data.csv", thousands=',')
df.drop(df.columns[0],axis=1,inplace=True)
df = df.dropna()
df = df[df['Type']=='Personal']
for col in ['Vehicle_Size','Driver_Age','Driver_Risk','Type']:
    le = LabelEncoder()
    le.fit(list(df[col].astype(str).values))
    df[col] = le.transform(list(df[col].astype(str).values))
df.head()
```

Out[2]:

	Year	Qtr	Vehicle_Size	Driver_Age	Driver_Risk	Type	Mileage	Bodily_Injury	Property_Damage
0	2009	1	2	0	2	0	15036	95	1538
1	2009	1	0	1	2	0	4563	21	390
2	2009	1	1	2	0	0	4322	82	481
3	2009	1	2	0	0	0	16801	96	1285
4	2009	1	1	1	2	0	13979	151	1086

In [4]:
```
#Choose all predictor variables
X = df.iloc[:,0:7]
X.head()
```

Out[4]:

	Year	Qtr	Vehicle_Size	Driver_Age	Driver_Risk	Type	Mileage
0	2009	1	2	0	2	0	15036
1	2009	1	0	1	2	0	4563
2	2009	1	1	2	0	0	4322
3	2009	1	2	0	0	0	16801
4	2009	1	1	1	2	0	13979

In [5]:
```
#Choose target variables individually
#bodily_injury
Y = df.iloc[:,7]
#property_damage
Y2 = df.iloc[:,8]
#comprehensive
Y3 = df.iloc[:,9]
#collision
Y4 = df.iloc[:,10]
#personal_injury
Y5 = df.iloc[:,11]
Y.head()
```

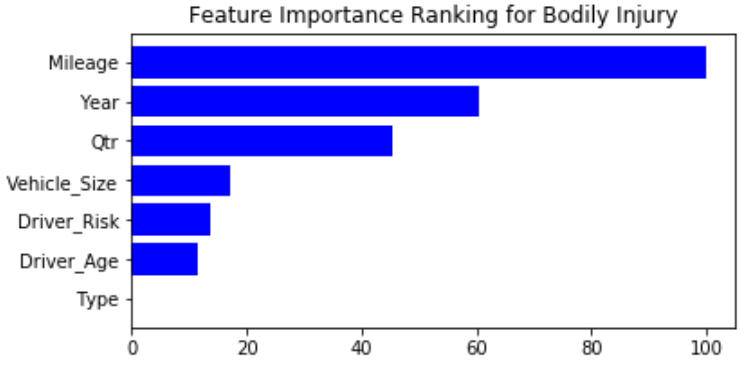
Out[5]:
```
0     95
1     21
2     82
3     96
4    151
Name: Bodily_Injury, dtype: int64
```

Feature Importance for each target variable

Target: Bodily Injury

In [6]:
```
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
importances = rf.feature_importances_


feature_importance = 100 * (importances/importances.max())#out of 100 percent importance
indices= np.argsort(feature_importance)
names = list(X_train.columns.values)
ordered_names = [names[i] for i in indices]
pos = np.arange(indices.shape[0]) + 0.5

plt.figure(figsize=(6,3))
colors = ['b']
plt.barh(pos, feature_importance[indices], align = 'center', color=colors)
plt.yticks(pos, ordered_names)
plt.title('Feature Importance Ranking for Bodily Injury')
plt.show()

print("Feature ranking:")

for f in range(X_train.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], feature_importance[indices[f]] ))
```

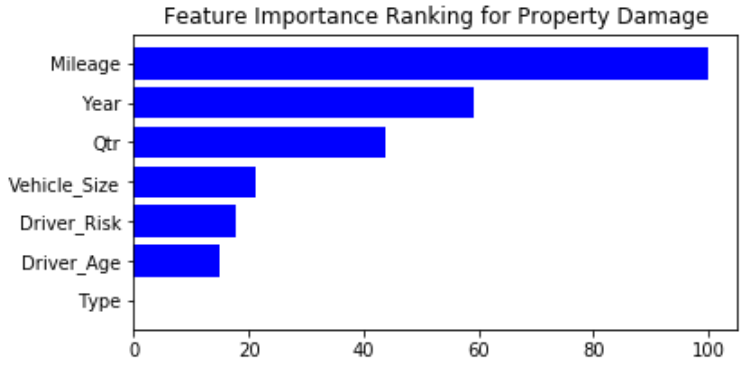
```
C:\Users\cyine\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWa
rning: This module was deprecated in version 0.18 in favor of the model_selection module
into which all the refactored classes and functions are moved. Also note that the interf
ace of the new CV iterators are different from that of this module. This module will be
removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
C:\Users\cyine\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: Depre
cationWarning: numpy.core.umath_tests is an internal NumPy module and should not be impo
rted. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```




```
Feature ranking:
1. feature 5 (0.000000)
2. feature 3 (11.581748)
3. feature 4 (13.698523)
4. feature 2 (17.137538)
5. feature 1 (45.347513)
6. feature 0 (60.301217)
7. feature 6 (100.000000)
```

Target: Property Damage

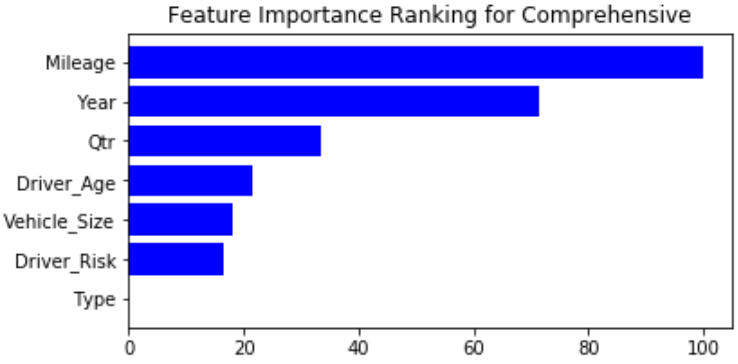
In [7]:
```
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, Y_train, Y_test = train_test_split(X, Y2, test_size=0.3)

rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
importances = rf.feature_importances_


feature_importance = 100 * (importances/importances.max())#out of 100 percent importance
indices= np.argsort(feature_importance)
names = list(X_train.columns.values)
ordered_names = [names[i] for i in indices]
pos = np.arange(indices.shape[0]) + 0.5

plt.figure(figsize=(6,3))
colors = ['b']
plt.barh(pos, feature_importance[indices], align = 'center', color=colors)
plt.yticks(pos, ordered_names)
plt.title('Feature Importance Ranking for Property Damage')
plt.show()

print("Feature ranking:")

for f in range(X_train.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], feature_importance[indices[f]] ))
```




```
Feature ranking:
1. feature 5 (0.000000)
2. feature 3 (14.955856)
3. feature 4 (17.650765)
4. feature 2 (21.202310)
5. feature 1 (43.667018)
6. feature 0 (59.070300)
7. feature 6 (100.000000)
```

Target: Comprehensive

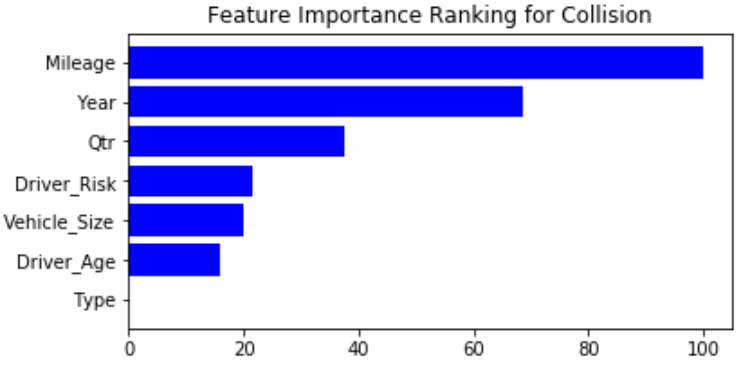
In [8]:
```
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, Y_train, Y_test = train_test_split(X, Y3, test_size=0.3)

rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
importances = rf.feature_importances_


feature_importance = 100 * (importances/importances.max())#out of 100 percent importance
indices= np.argsort(feature_importance)
names = list(X_train.columns.values)
ordered_names = [names[i] for i in indices]
pos = np.arange(indices.shape[0]) + 0.5

plt.figure(figsize=(6,3))
colors = ['b']
plt.barh(pos, feature_importance[indices], align = 'center', color=colors)
plt.yticks(pos, ordered_names)
plt.title('Feature Importance Ranking for Comprehensive')
plt.show()

print("Feature ranking:")

for f in range(X_train.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], feature_importance[indices[f]] ))
```




```
Feature ranking:
1. feature 5 (0.000000)
2. feature 4 (16.650424)
3. feature 2 (18.173295)
4. feature 3 (21.541093)
5. feature 1 (33.611248)
6. feature 0 (71.417882)
7. feature 6 (100.000000)
```

Target: Collision

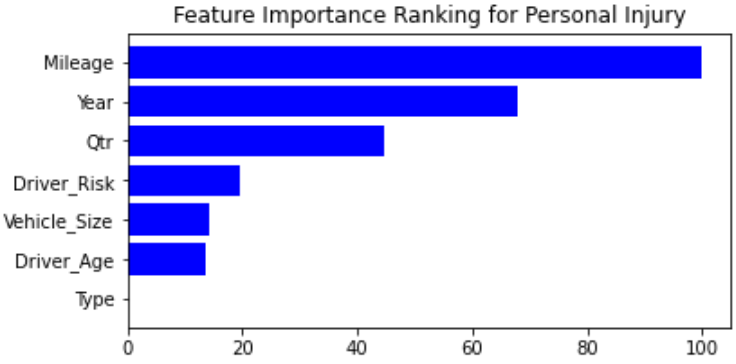
In [9]:
```
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, Y_train, Y_test = train_test_split(X, Y4, test_size=0.3)

rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
importances = rf.feature_importances_


feature_importance = 100 * (importances/importances.max())#out of 100 percent importance
indices= np.argsort(feature_importance)
names = list(X_train.columns.values)
ordered_names = [names[i] for i in indices]
pos = np.arange(indices.shape[0]) + 0.5

plt.figure(figsize=(6,3))
colors = ['b']
plt.barh(pos, feature_importance[indices], align = 'center', color=colors)
plt.yticks(pos, ordered_names)
plt.title('Feature Importance Ranking for Collision')
plt.show()

print("Feature ranking:")

for f in range(X_train.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], feature_importance[indices[f]] ))
```




```
Feature ranking:
1. feature 5 (0.000000)
2. feature 3 (15.937080)
3. feature 2 (19.921890)
4. feature 4 (21.433976)
5. feature 1 (37.546780)
6. feature 0 (68.468958)
7. feature 6 (100.000000)
```

Target: Personal Injury

In [100]:
```
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, Y_train, Y_test = train_test_split(X, Y5, test_size=0.3)

rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
importances = rf.feature_importances_


feature_importance = 100 * (importances/importances.max())#out of 100 percent importance
indices= np.argsort(feature_importance)
names = list(X_train.columns.values)
ordered_names = [names[i] for i in indices]
pos = np.arange(indices.shape[0]) + 0.5

plt.figure(figsize=(6,3))
colors = ['b']
plt.barh(pos, feature_importance[indices], align = 'center', color=colors)
plt.yticks(pos, ordered_names)
plt.title('Feature Importance Ranking for Personal Injury')
plt.show()

print("Feature ranking:")

for f in range(X_train.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], feature_importance[indices[f]] ))
```




```
Feature ranking:
1. feature 5 (0.000000)
2. feature 3 (13.737128)
3. feature 2 (14.266700)
4. feature 4 (19.514573)
5. feature 1 (44.618397)
6. feature 0 (67.803047)
7. feature 6 (100.000000)
```

From the feature importance plots for each target variable, it can be suggested that Mileage and Year are relatively important variables that may be used for modeling.

In [ ]:
```
!export PATH=/Library/TeX/texbin:$PATH
```