CSXXX ASSIGNMENT-X REPORT

Title of the Report

Deniz Gokcin

November 18, 2020

Introduction

This report explains the implementation details of CS552 Data Science with Python, Assignment 1, which is about predicting the car prices using linear regression. I will first briefly give some information about linear regression, than I will describe my implementation details. I will continue with the results by evaluating the trained models with different metrics and lastly, I will conclude by discussing on the extracted information.

To run the code, please make sure that all the requirements in requirements.txt file is installed on your environment.

Linear Regression

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Implementation Details

Feature Engineering

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[1638]:
```
# Convert year to age
df = df.rename(columns={"year": "age"})
df["age"] = dt.now().year - df["age"]
```

[1639]:
```
# Convert car name to brand
df['car_brand'] = df['name'].apply(lambda x:x.split(' ')[0])
```

[1640]:
```
# Drop the name column, since it is not needed anymore
df.drop(['name'], axis=1, inplace=True)
```

[1641]:
```
# Get rid of units
df['mileage'] = df['mileage'].str.extract('(\d+)', expand=False)
df['engine'] = df['engine'].str.extract('(\d+)', expand=False)
df['max_power'] = df['max_power'].str.extract('(\d+)', expand=False)

# Need to drop na liens again since there is now na cells.
df.dropna(inplace=True)
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[1642]:
```
# One-hot encoding for categorical features
df = pd.concat([df,pd.get_dummies(df['fuel'], prefix='fuel')],axis=1)
df = pd.concat([df,pd.get_dummies(df['seller_type'], prefix='seller')],axis=1)
df = pd.concat([df,pd.get_dummies(df['owner'], prefix='owner')],axis=1)
df = pd.concat([df,pd.get_dummies(df['car_brand'], prefix='brand')],axis=1)
df = pd.concat([df,pd.get_dummies(df['transmission'], prefix='transmission')],
               axis=1)
```

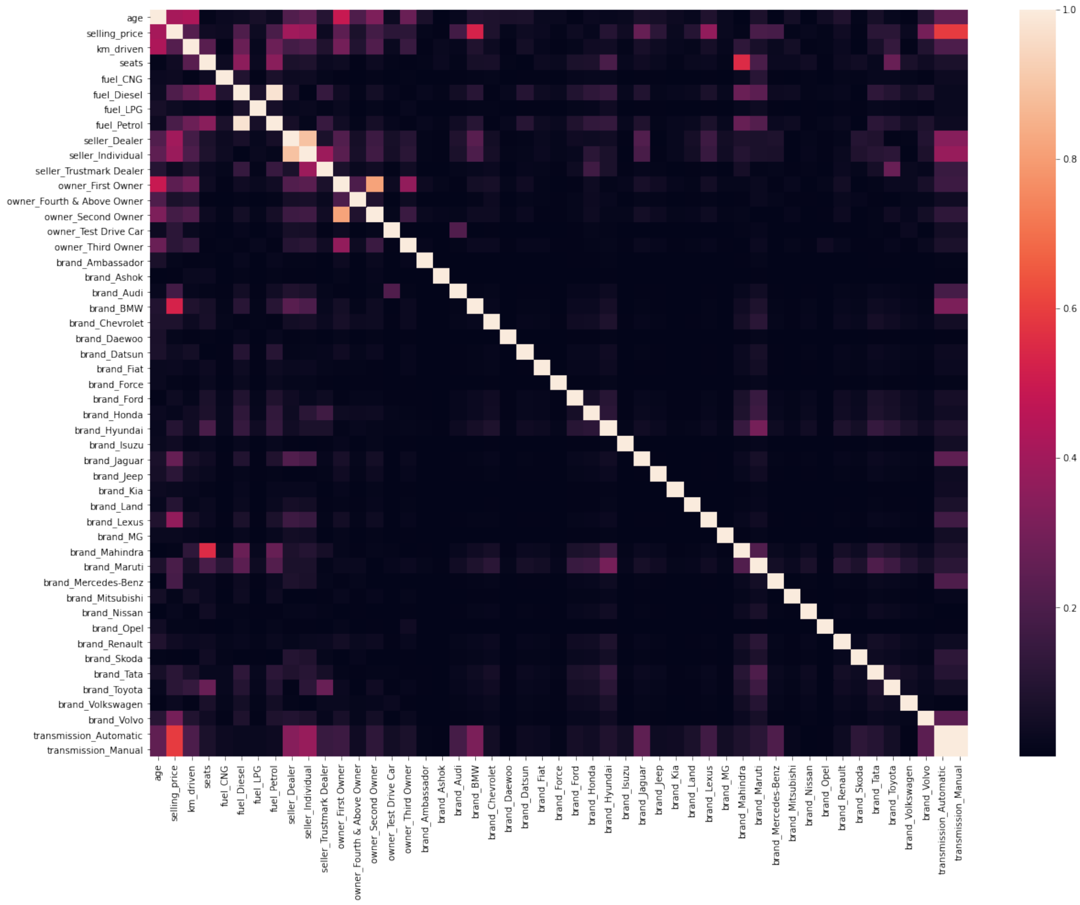
[1643]:
```
# Drop the pre-one-hot columns
df.drop(['fuel'], axis=1, inplace=True)
df.drop(['seller_type'], axis=1, inplace=True)
df.drop(['owner'], axis=1, inplace=True)
df.drop(['transmission'], axis=1, inplace=True)
df.drop(['car_brand'], axis=1, inplace=True)
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[1644]:
```
# Pop selling price for later use
selling_price = df.pop('selling_price')
```

[1647]:
```
# Visualize the correlation using seaborn
sns.heatmap(corr_matrix)
```

[1647]: <AxesSubplot:>



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[1648]:
```python
# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.
 ↪bool))

# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.98)] +\
          [column for column in upper.columns if any(upper[column] < 0.01)]

# to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]
# Print what will be dropped
print("The columns to be dropped: " + str(to_drop))
```

```
The columns to be dropped: ['transmission_Manual', 'seats', 'fuel_CNG',
'fuel_LPG', 'seller_Individual', 'owner_First Owner', 'owner_Fourth & Above
Owner', 'owner_Second Owner', 'owner_Test Drive Car', 'owner_Third Owner',
'brand_Ambassador', 'brand_Ashok', 'brand_Audi', 'brand_BMW', 'brand_Chevrolet',
'brand_Daewoo', 'brand_Datsun', 'brand_Fiat', 'brand_Force', 'brand_Ford',
'brand_Honda', 'brand_Hyundai', 'brand_Isuzu', 'brand_Jaguar', 'brand_Jeep',
'brand_Kia', 'brand_Land', 'brand_Lexus', 'brand_MG', 'brand_Mahindra',
'brand_Maruti', 'brand_Mercedes-Benz', 'brand_Mitsubishi', 'brand_Nissan',
'brand_Opel', 'brand_Renault', 'brand_Skoda', 'brand_Tata', 'brand_Toyota',
'brand_Volkswagen', 'brand_Volvo', 'transmission_Automatic',
'transmission_Manual']
```

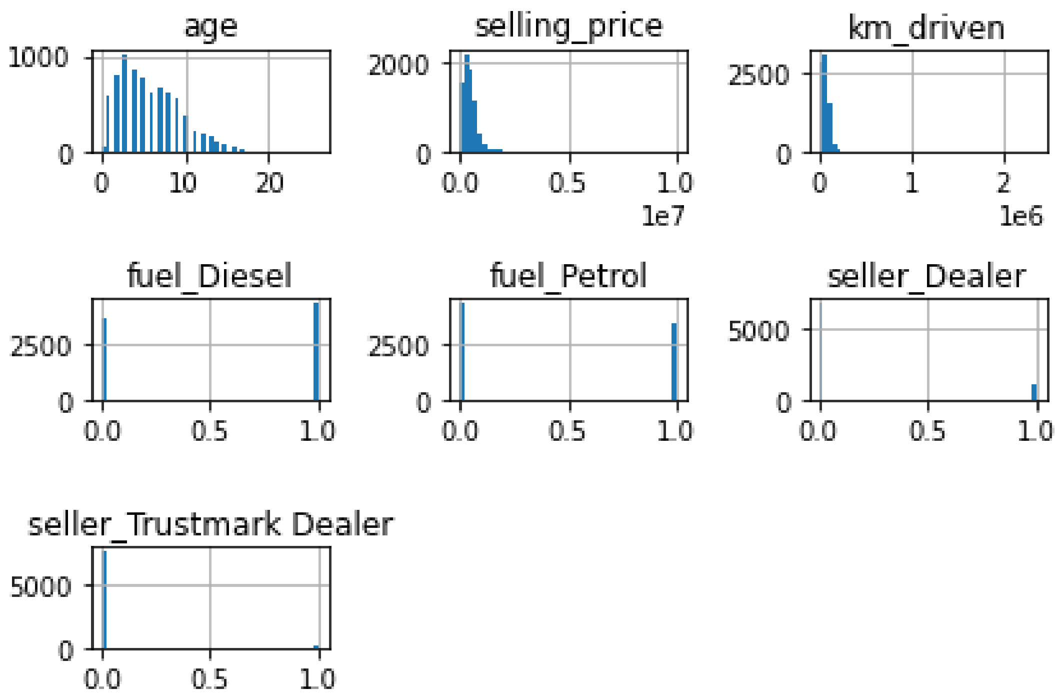
[1393]:
```python
    # Check the data
    df.head()
```

[1393]:
```
   age  selling_price  km_driven mileage engine max_power  fuel_Diesel  \
0    6         450000     145500      23   1248        74            1
1    6         370000     120000      21   1498       103            1
2   14         158000     140000      17   1497        78            0
3   10         225000     127000      23   1396        90            1
4   13         130000     120000      16   1298        88            0

   fuel_Petrol  seller_Dealer  seller_Trustmark Dealer
0            0              0                        0
1            0              0                        0
2            1              0                        0
3            0              0                        0
4            1              0                        0
```

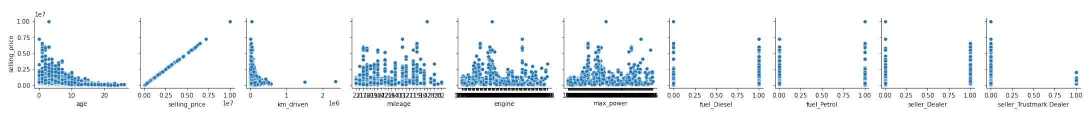
I visualized the remaining columns in the dataframe with a histogram with the following snippet.

[1394]:
```
# Draw histogram of the data for visualization
hist = df.hist(bins=50)
plt.tight_layout()
```



[1395]:
```
# Visualizing the data for each dependent feature with independent feature
pos_of_col = df.columns.get_loc("selling_price") + 1
x_vars = df.iloc[:,np.r_[range(pos_of_col-1),range(pos_of_col,len(df.
↪columns)
                                                      )]].columns
x_vars = df.columns

sns.pairplot(data=df,
             y_vars=['selling_price'],
             x_vars=x_vars,
             diag_kind=None
             )
plt.tight_layout()
```



Training the Data

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[1396]:
```
# Training the data
```

[1397]:
```
df.head()
```

[1397]:
```
   age  selling_price  km_driven  mileage  engine  max_power  fuel_Diesel  \
0    6         450000     145500       23    1248         74            1
1    6         370000     120000       21    1498        103            1
2   14         158000     140000       17    1497         78            0
3   10         225000     127000       23    1396         90            1
4   13         130000     120000       16    1298         88            0

   fuel_Petrol  seller_Dealer  seller_Trustmark Dealer
0            0              0                        0
1            0              0                        0
2            1              0                        0
3            0              0                        0
4            1              0                        0
```

[1398]:
```
# Generate different featuresets and save the used columns to a variable␣
↪for
# later pickleing.
feature_set = df[['age', 'max_power', 'fuel_Diesel', 'km_driven']]
# feature_set = df[['age', 'max_power', 'fuel_Diesel', 'engine']]
# feature_set = df[['age', 'mileage', 'fuel_Diesel', 'km_driven']]
feature_set_cols = '-'.join(list(feature_set.columns))
```

[1399]:
```
# x = df.iloc[:,0:-1].values.astype(float)
y = df['selling_price'].astype(float)
```

[1400]:
```
# Split the data to training and test sets
x_train, x_test, y_train, y_test = train_test_split(feature_set, y,␣
↪test_size=0.2,
                                                    random_state=147)
```

[1401]:
```
# Initialize min-max scaler and transform each feature by using min-max␣
↪scaler
# You need to put the feature values to a certain range (in general: (0,␣
↪1)) in order to stabilize the model
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
x_train = scaler.fit_transform(x_train)
```

[1402]:
```
# Print the number of instances in training & test set
print(x_train.shape)
print(x_test.shape)
```

```
(6324, 4)
(1582, 4)
```

[1403]:
```
# Initialize the linear regression model
model = LinearRegression()
```

[1405]:
```
# Print the general formula of our linear regression model
_str = "y = "
for i, m in enumerate(model.coef_):
    _str += "x_{}*{}+".format(i+1, m)
_str += str(model.intercept_)
print(_str)
```

```
y = x_1*-1202121.4876479814+x_2*5896958.923500527+x_3*8632.
↪784887891676+x_4*-335
0152.137738197+73752.16816291772
```

[1406]:
```
# Scale each feature to range(0, 1)
x_test = scaler.transform(x_test)
```

[1408]:
```
# Predict the values by using all test data
y_pred = model.predict(x_test)
```

Results

[1410]:
```
# Calculate the score of the model in test data
score = model.score(x_test, y_test)
print(score)
```

```
0.6417537167824829
```

[1411]:
```
# Calculate mean squared error of predicted values
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
195623662859.41428
```

[1412]:
```
# Calculate absolute squared error of predicted values
mae = mean_absolute_error(y_test, y_pred)
print(mae)
```

```
275135.5069473646
```

[1413]:

```
# Calculate  rsquared error of predicted values
r2e = r2_score(y_test, y_pred)
print(r2e)
```

```
0.6417537167824829
```

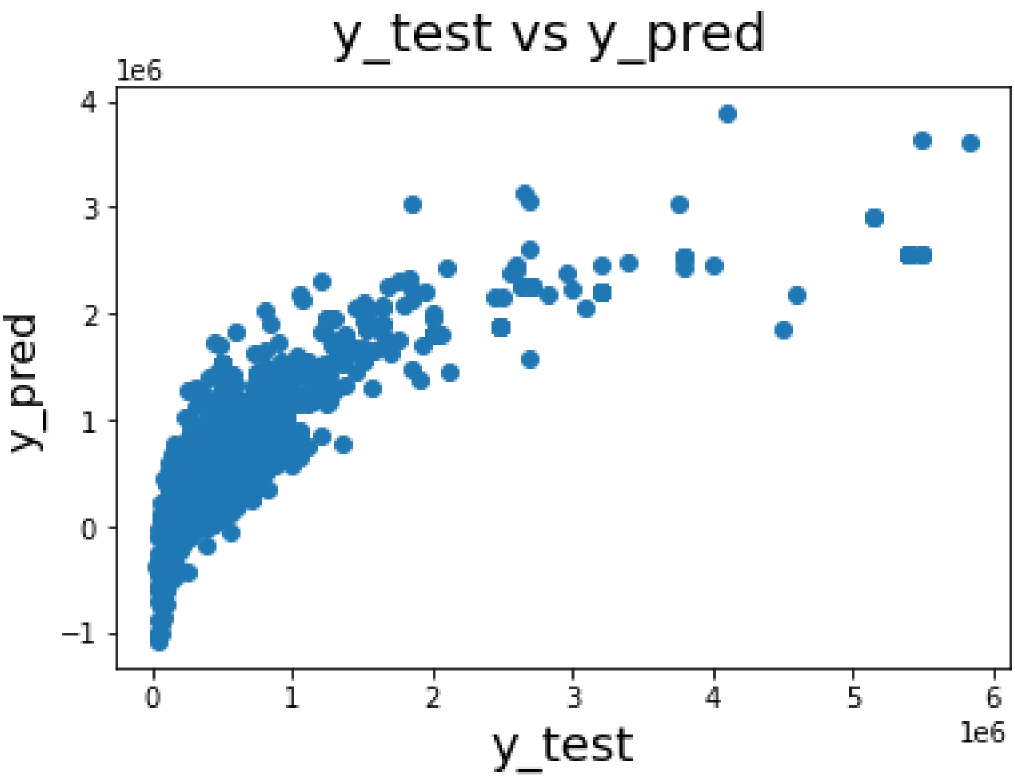
[1414]:

```
# Calculate  root mean squared error of predicted values
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(rmse)
```

```
195623662859.41428
```

	Mean Squared Error	Mean Absolute Error	R Squared Error	Root Mean Squared Error	Accuracy
['age', 'max_power', 'fuel_Diesel', 'km_driven']	195623662859.41428	275135.5069473646	0.6417537167824829	442293.63872818055	0.6417537167824829
['age', 'max_power', 'fuel_Diesel', 'engine']	201881726980.0854	279069.76902061084	0.6302933025432378	449312.5048116126	0.6302933025432378
['age', 'mileage', 'fuel_Diesel', 'km_driven']	362999740519.1677	335505.19209081476	0.335237333004182	602494.5979170002	0.335237333004182

[1415]:
```
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)
```



Conclusions

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Bibliography