**IBM Data Science Professional Certificate**

**Applied Data Science Capstone Project**

# Predicting Severity of Car Collisions in Seattle

By: Rama Chaitanya Samanchi

September 2020

# ➢ Introduction

Driving a car is important today as it gives people power personal control and autonomy. So, the project deals with an aspect that is interesting from a Data Analyst's perspective while we look at the safety associated with driving a car. It would be great if we can model collisions associated with cars which can aid in understanding the factors associated with a collision. But what is the Business Interest involved in such a study? What are the factors which are important for the study? The following two sections will answer these questions to kick-start our project (with safety!).

## *Phase-1: Business Understanding*

**Problem Statement**

The objective of the project is to develop a model that can predict the severity of a car collision based on several important factors that influence a collision. The study requires a classification model that can predict the impact of an accident severity in a metro city called Seattle, Washington.

**Background**

For each accident taken place, the local traffic department of Seattle records each incident with a unique key and multiple factors related to the collision like the location, the number of people involved in, weather conditions, road conditions, light conditions, speeding, etc. These details are some of the important details which contribute to the severity of an accident. Certainly, these are the factors to investigate and analyze the data to build a model that can predict the severity of the collision to start with. The study will also provide the factors by importance in predicting the severity.

**Target Stakeholders**

The model is very helpful for the traffic government to take mitigative actions to reduce car collisions based on the factors which have a large influence in predicting the severity of the collision. For example, the traffic department can take preventive actions if observed that road condition is playing an important role in the classifying the severity of an incident. They can not only repair the road on which current accidents have taken place but also take preventive action by identifying such roads in similar condition and fix them to reduce the possibility of incidents in the future. Similarly, Smart Street Lighting Systems can be deployed to improve the visibility on the streets if Lighting condition was found to have a significant role in the severity of the collision. Therefore, the administration is the first and foremost target stakeholders of the project.

Also, the emergency service provider and healthcare systems will have an effective tool in predicting the severity of the collision to provide necessary aid to potentially save the lives and improve the life-sensitivity. They are second target stakeholders to get benefitted by the model as an accurate model is handy in predicting the severity in taking proactive action.

## ➢ Data

The dataset is recorded by the traffic department of Seattle and available on the government site which is provided through Coursera. We were given an option either to choose the one provided or select a new one. I am going ahead with the one provided by Coursera for the simplicity and effective building of the model.

### *Phase 2: Data Understanding*

**Dataset Description**

The dataset is recorded by the traffic department of Seattle and available on the government site which is provided through Coursera. We were given an option either to choose the one provided or select a new one. I am going ahead with the one provided by Coursera for the simplicity and effective building of the model. The dataset contains 194673 rows as the dataset captured the collision for multiple years i.e. from 2004 to 2020 (May). The dataset has 37 columns which include description columns as well which will be dropped while building the model.

**Columns**

| Column Name | Data Type | Description |
|---|---|---|
| OBJECTID | ObjectID | ESRI unique identifier |
| SHAPE | Geometry | ESRI geometry field |
| INCKEY | Long | A unique key for the incident |
| COLDETKEY | Long | Secondary key for the incident |
| ADDRTYPE | Text, 12 | Collision address type: • Alley • Block • Intersection |
| INTKEY | Double | Key that corresponds to the intersection associated with a collision |
| LOCATION | Text, 255 | Description of the general location of the collision |
| EXCEPTRSNCODE | Text, 10 | |
| EXCEPTRSNDESC | Text, 300 | |
| SEVERITYCODE | Text, 100 | A code that corresponds to the severity of the collision: • 3—fatality • 2b—serious injury • 2—injury • 1—prop damage • 0—unknown |
| SEVERITYDESC | Text | A detailed description of the severity of the collision |
| COLLISIONTYPE | Text, 300 | Collision type |
| PERSONCOUNT | Double | The total number of people involved in the collision |
| PEDCOUNT | Double | The number of pedestrians involved in the collision. This is entered by the state |
| PEDCYLCOUNT | Double | The number of bicycles involved in the collision. This is entered by the state |
| VEHCOUNT | Double | The number of vehicles involved in the collision. This is entered by the state |
| INJURIES | Double | The number of total injuries in the collision. This is entered by the state |
| SERIOUSINJURIES | Double | The number of serious injuries in the collision. This is entered by the state |
| FATALITIES | Double | The number of fatalities in the collision. This is entered by the state |
| INCDATE | Date | The date of the incident. INCDTTM Text, 30 The date and time of the incident. |
| JUNCTIONTYPE | Text, 300 | Category of the junction at which collision took place |
| SDOT_COLCODE | Text, 10 | A code is given to the collision by SDOT. |

| | | |
|---|---|---|
| SDOT_COLDESC | Text, 300 | A description of the collision corresponding to the collision code. |
| INATTENTIONIND | Text, 1 | Whether or not collision was due to inattention. (Y/N) |
| UNDERINFL | Text, 10 | Whether or not a driver involved was under the influence of drugs or alcohol |
| WEATHER | Text, 300 | A description of the weather conditions during the time of the collision. |
| ROADCOND | Text, 300 | The condition of the road during the collision. |
| LIGHTCOND | Text, 300 | The light conditions during the collision. |
| PEDROWNOTGRNT | Text, 1 | Whether or not the pedestrian right of way was not granted. (Y/N) |
| SDOTCOLNUM | Text, 10 | A number is given to the collision by SDOT. |
| SPEEDING | Text, 1 | Whether or not speeding was a factor in the collision. (Y/N) |
| ST_COLCODE | Text, 10 | A code provided by the state that describes the collision. For more information about these codes, please see the State Collision Code Dictionary. |
| ST_COLDESC | Text, 300 | A description that corresponds to the state's coding designation. |
| SEGLANEKEY | Long | A key for the lane segment in which the collision occurred. |
| CROSSWALKKEY | Long | A key for the crosswalk at which the collision occurred. |
| HITPARKEDCAR | Text, 1 | Whether or not the collision involved hitting a parked car. (Y/N) |

**Key pattern observations**

1. These are the columns and their number of null values.

```
Data_Collisions.isnull().sum()
```

```
SEVERITYCODE           0
X                   5334
Y                   5334
OBJECTID               0
INCKEY                 0
COLDETKEY              0
REPORTNO               0
STATUS                 0
ADDRTYPE            1926
INTKEY            129603
LOCATION            2677
EXCEPTRSNCODE     109862
EXCEPTRSNDESC     189035
SEVERITYCODE.1         0
SEVERITYDESC           0
COLLISIONTYPE       4904
PERSONCOUNT            0
PEDCOUNT               0
PEDCYLCOUNT            0
VEHCOUNT               0
INCDATE                0
INCDTTM                0
JUNCTIONTYPE        6329
SDOT_COLCODE           0
SDOT_COLDESC           0
INATTENTIONIND    164868
UNDERINFL           4884
WEATHER             5081
ROADCOND            5012
LIGHTCOND           5170
PEDROWNOTGRNT     190006
SDOTCOLNUM         79737
SPEEDING          185340
ST_COLCODE            18
ST_COLDESC          4904
SEGLANEKEY             0
CROSSWALKKEY           0
HITPARKEDCAR           0
dtype: int64
```

2. Dropped some columns which are descriptions and have too many null values.

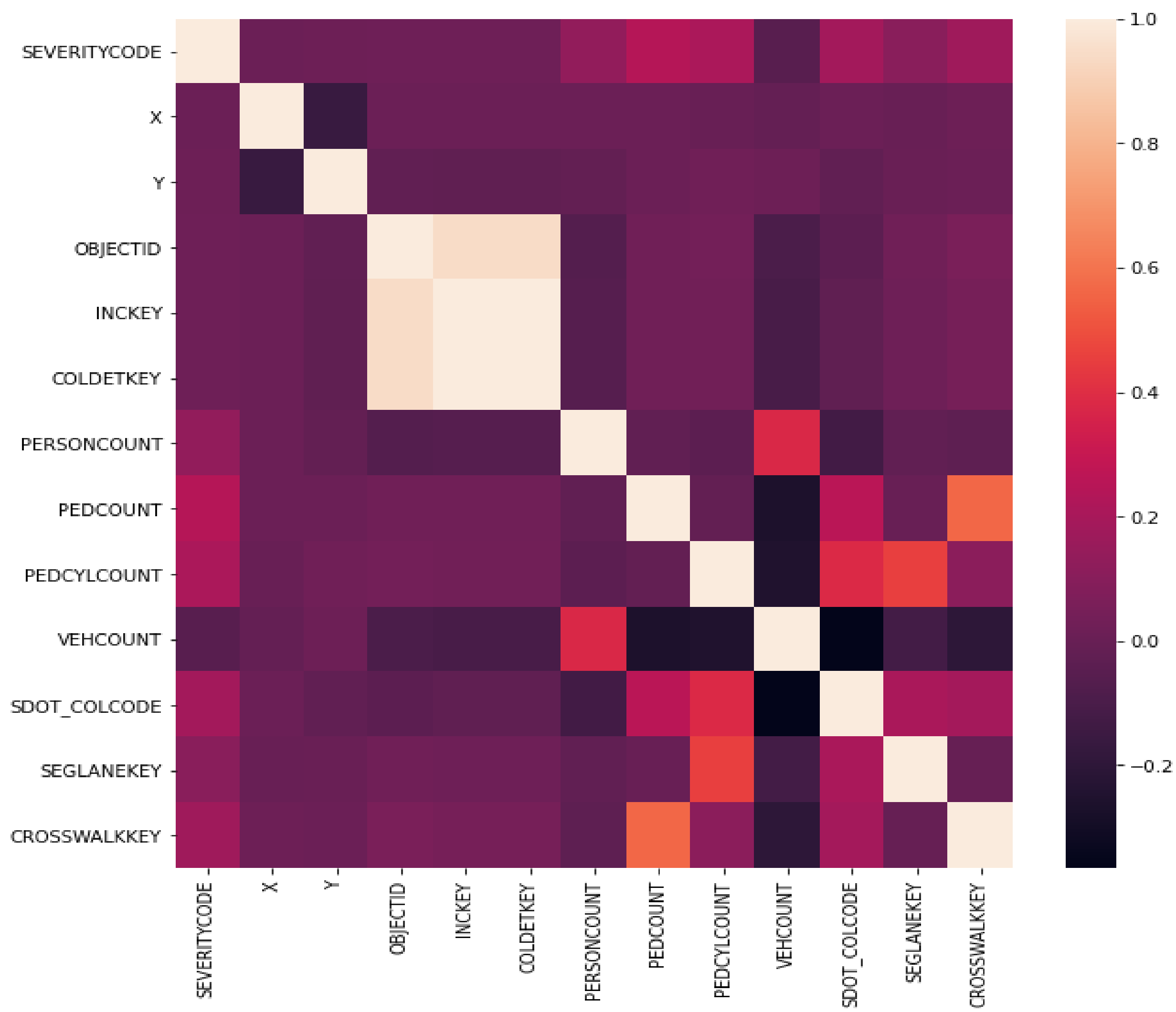
```
New_Collisions = Data_Collisions.drop(['INTKEY', 'EXCEPTRSNCODE', 'SEVERITYCODE.1',
                                       'EXCEPTRSNDESC', 'SEVERITYDESC', 'SDOT_COLDESC',
                                       'INATTENTIONIND', 'PEDROWNOTGRNT','SDOTCOLNUM',
                                       'SPEEDING', 'ST_COLDESC'], axis=1)
```

3. The Target variable Severity Code is imbalanced with two classifications 1 & 2.

```
New_Collisions['SEVERITYCODE'].value_counts()
```

```
1    136485
2     58188
Name: SEVERITYCODE, dtype: int64
```

4. An early correlation heat suggests that there is not much correlation with some columns missing due to text datatype.



## *Phase 3: Data Preparation*

Data Preparation is the most crucial phase of a Data Science project, as the data frame on which the model is going to be built will be prepared in this phase. Data is often messy and needs to be cleaned up to facilitate analysis. The given dataset contains 194673 rows with 37 columns. But we certainly came across data points which cannot help in making meaningful analysis.

So, this phase is further divided into:

- Selection of columns

```
SEVERITYCODE    Target Variable
X               Dropped, Coordinate
Y               Dropped, Coordinate
OBJECTID        Dropped, ID Field
INCKEY          Dropped, ID Field
COLDETKEY       Dropped, ID Field
REPORTNO        Dropped, ID Field
STATUS          Dropped, Not found in Meta Data
ADDRTYPE        Selected
INTKEY          Dropped, Too many null values
LOCATION        Dropped, Location not required
EXCEPTRSNCODE   Dropped, only 1 category
EXCEPTRSNDESC   Dropped, the Description column
SEVERITYCODE.1  Dropped, Duplicate Column
SEVERITYDESC    Dropped, the Description column
COLLISIONTYPE   Selected
PERSONCOUNT     Selected
PEDCOUNT        Selected
PEDCYLCOUNT     Selected
VEHCOUNT        Selected
INCDATE         Dropped, Date Field not Required
INCDTTM         Dropped, Date Field not Required
JUNCTIONTYPE    Selected
SDOT_COLCODE    Selected
SDOT_COLDESC    Dropped, the Description column
INATTENTIONIND  Dropped, only 1 category
UNDERINFL       Selected
WEATHER         Selected
ROADCOND        Selected
LIGHTCOND       Selected
PEDROWNOTGRNT   Dropped, Too many null values
SDOTCOLNUM      Dropped, Too many null values
SPEEDING        Dropped, Too many null values
ST_COLCODE      Selected
ST_COLDESC      Dropped, the Description column
SEGLANEKEY      Most of the values are 0, insignificant
CROSSWALKKEY    Most of the values are 0, insignificant
HITPARKEDCAR    Selected
```

So, the columns which are finally selected are:
SEVERITYCODE, ADDRTYPE, COLLISIONTYPE, PERSONCOUNT, PEDCOUNT, PEDCYLCOUNT, VEHCOUNT , JUNCTIONTYPE, SDOT_COLCODE, UNDERINFL, WEATHER, ROADCOND, LIGHTCOND, ST_COLCODE, HITPARKEDCAR

- Dropping null values

Columns with a reasonable number of null values, ADDRTYPE, COLLISIONTYPE, JUNCTIONTYPE, UNDERINFL, WEATHER, ROADCOND, LIGHTCOND, ST_COLCODE, were cleaned up by dropping the null values. As all the columns are categorical, it is tough to replace the values by mode, which has a higher chance of misinterpretation of truth. So, it is better to get rid of null values. Moreover, dropping 4000-6000 rows from a dataset of 190000+ rows is the best way to clean up the data frame as it is taking away less than 3% of the rows. The number of null values before cleaning up the null values and the number of null values after cleaning up the null values are shown below:

```
New_Collisions.isnull().sum()
```

```
SEVERITYCODE        0
ADDRTYPE         1926
COLLISIONTYPE    4904
PERSONCOUNT         0
PEDCOUNT            0
PEDCYLCOUNT         0
VEHCOUNT            0
JUNCTIONTYPE     6329
SDOT_COLCODE        0
UNDERINFL        4884
WEATHER          5081
ROADCOND         5012
LIGHTCOND        5170
ST_COLCODE         18
HITPARKEDCAR        0
dtype: int64
```

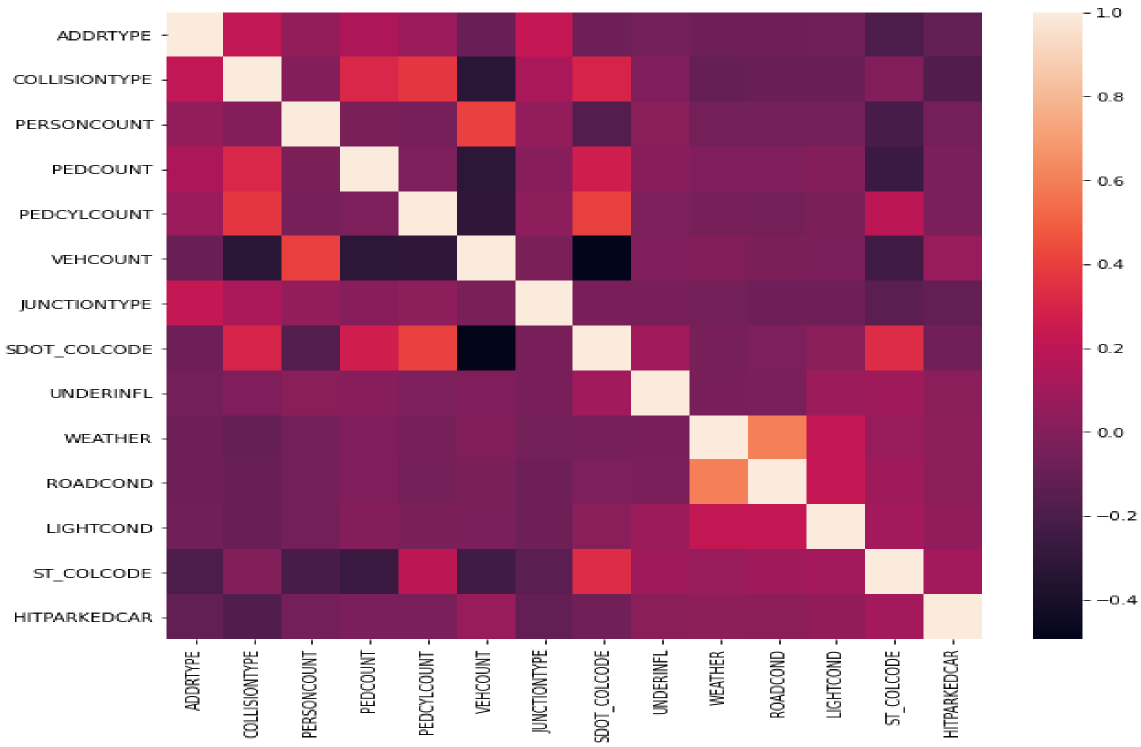
```
New_Collisions1 = New_Collisions.dropna(subset=['ADDRTYPE', 'COLLISIONTYPE', 'JUNCTIONTYPE',
                                               'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND', 'ST_COLCODE'])
```

```
New_Collisions1.isnull().sum()
```

```
SEVERITYCODE     0
ADDRTYPE         0
COLLISIONTYPE    0
PERSONCOUNT      0
PEDCOUNT         0
PEDCYLCOUNT      0
VEHCOUNT         0
JUNCTIONTYPE     0
SDOT_COLCODE     0
UNDERINFL        0
WEATHER          0
ROADCOND         0
LIGHTCOND        0
ST_COLCODE       0
HITPARKEDCAR     0
dtype: int64
```

- Plotting heatmap for correlation between variables

All the 14 columns are plotted against each other on a heat map to check the correlation between them. The correlation is < 0.6 for all the combinations except for the diagonal comparisons, which means that there is no problem of multicollinearity in the dataset.




- Categorize some columns

All the categorical values of the columns are converted on a numerical scale for algorithmic purposes.

```
In [25]:  Cleanup = {"ADDRTYPE":      {"Block": 1, "Intersection": 2, "Alley": 3},
                     "COLLISIONTYPE": {"Parked Car": 1, "Angles": 2, "Rear Ended": 3, "Other": 4,
                                       "Sideswipe": 5, "Left Turn": 6, "Pedestrian": 7, "Cycles": 8, "Right Turn": 9, "Head On": 10},
                     "JUNCTIONTYPE":  {"Mid-Block (not related to intersection)": 1, "At Intersection (intersection related)": 2,
                                       "Mid-Block (but intersection related)": 3, "Driveway Junction": 4,
                                       "At Intersection (but not related to intersection)": 5, "Ramp Junction": 6, "Unknown": 7},
                     "UNDERINFL":     {"Y": 1, "N":0},
                     "WEATHER":       {"Clear": 1, "Raining": 2, "Overcast": 3, "Unknown": 4, "Snowing": 5, "Other": 6,
                                       "Fog/Smog/Smoke": 7, "Sleet/Hail/Freezing Rain": 8, "Blowing Sand/Dirt": 9,
                                       "Severe Crosswind": 10, "Partly Cloudy": 11},
                     "ROADCOND":      {"Dry": 1, "Wet": 2, "Unknown": 3, "Ice": 4, "Snow/Slush": 5, "Other": 6, "Standing Water": 7,
                                       "Sand/Mud/Dirt": 8, "Oil":9},
                     "LIGHTCOND":     {"Daylight": 1, "Dark - Street Lights On": 2, "Unknown": 3, "Dusk": 4, "Dawn": 5,
                                       "Dark - No Street Lights": 6, "Dark - Street Lights Off": 7, "Other": 8,
                                       "Dark - Unknown Lighting": 9},
                     "HITPARKEDCAR":  {"Y": 1, "N":0}}
```

- Change of Data Types

The datatypes of UNDERINFL & ST_COLCODE are converted into the integer type.

```
In [73]:  New_Collisions1['UNDERINFL'] = np.int64(New_Collisions1['UNDERINFL'])
          New_Collisions1['ST_COLCODE'] = np.int64(New_Collisions1['ST_COLCODE'])
```

- Balancing the dataset to avoid overfitting issues

The Dataset needs to be balanced because the Target variable SEVERITYCODE is unbalanced across its categories. SMOTE Technique is employed to the same. The Severity Code is defined as y (target variable) & the rest of the variables are assigned as the independent variables. The Training & Testing dataset split is set to 75:25. This split is assigned to the SMOTE Sampling Strategy of 1.0.

```
# Balancing the data with SMOTE Technique, Splitting the Training & Test Data

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

y = New_Collisions1['SEVERITYCODE']
X = New_Collisions1.drop('SEVERITYCODE', axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=27)

sm = SMOTE(random_state=27, sampling_strategy=1.0)
X_train, y_train = sm.fit_sample(X_train, y_train)
```

## ➢ <u>Methodology</u>

### *Phase 4: Modeling*

- Modeling without PCA

Since, this is a classification problem statement, I have chosen 4 classification models: Decision Tree, Random Forest, Naïve Bayes, & KNN Neighbors to model the data and check the accuracy of each. This modeling approach is conducted before performing Principal Component Analysis with all the 14 columns. From the results below, it is evident that the KNN Neighbors model stands the best with an accuracy of 71.73%.

```
print("Decision Tree's Accuracy score:", round(Decision_Tree_Accuracy_without_PCA, 2))
print("Random Forest's Accuracy score:", round(Random_Forest_without_PCA, 2))
print("Naive Bayes Classifiers's Accuracy score:", round(Naïve_Bayes_without_PCA, 2))
print("KNN Neighbors Accuracy score:", round(KNN_Neighbors_without_PCA, 2))
```

```
Decision Tree's Accuracy score: 67.1
Random Forest's Accuracy score: 67.63
Naive Bayes Classifiers's Accuracy score: 71.36
KNN Neighbors Accuracy score: 71.73
```

- Principal Component Analysis

For an explained variance of 95%, the number of features reduced from 14 to 13. What the features are will be explained in the Evaluation phase.

```
reduced.explained_variance_ratio_
```

```
array([0.17603498, 0.14489198, 0.10777555, 0.08147445, 0.07689367,
       0.06957225, 0.06590487, 0.05855866, 0.05441824, 0.0440079 ,
       0.03949239, 0.02867866, 0.02853283])
```

- Modeling with PCA

Models are generated on the dataset after PCA. From the results below, Random Forest has the highest accuracy score of 74.05%. This is the best model out of all the models presented so far.

```
print("Decision Tree's Accuracy score after PCA:", round(Decision_Tree_with_PCA, 2))
print("Random Forest's Accuracy score after PCA:", round(Random_Forest_with_PCA, 2))
print("Naive Bayes Classifiers's Accuracy score after PCA:", round(Naive_Bayes_with_PCA, 2))
print("KNN Neighbors Accuracy score after PCA:", round(KNN_Neighbors_with_PCA, 2))
```

```
Decision Tree's Accuracy score after PCA: 73.56
Random Forest's Accuracy score after PCA: 74.05
Naive Bayes Classifiers's Accuracy score after PCA: 70.86
KNN Neighbors Accuracy score after PCA: 73.89
```

## *Phase 5: Evaluation*

- Feature Importance

The next question we are trying to answer in the analysis is the three most important factors in the model which can explain the variance. We have conducted feature importance on the model with the highest accuracy, i.e. the model with 74.05% accuracy. As HITPARKEDCAR is dropped after conducting PCA, it is not shown in the result. From the results below, ADDRTYPE, PEDCOUNT, & COLLISIONTYPE are the 3 most important features, which are explaining a 40% variance together.

```
#Q2- What are the three most important features in this model.
import pandas as pd
cols = ['ADDRTYPE', 'COLLISIONTYPE', 'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'JUNCTIONTYPE',
        'SDOT_COLCODE', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND', 'ST_COLCODE', 'HITPARKEDCAR']
feature_importances=pd.DataFrame(PCA_RF_model.feature_importances_,index =cols[0:13],
columns=['importance']).sort_values('importance',ascending=False)
feature_importances
```

| | importance |
|---|---|
| ADDRTYPE | 0.179602 |
| PEDCOUNT | 0.112927 |
| COLLISIONTYPE | 0.111116 |
| ROADCOND | 0.093769 |
| PEDCYLCOUNT | 0.081783 |
| WEATHER | 0.072709 |
| JUNCTIONTYPE | 0.066165 |
| PERSONCOUNT | 0.063679 |
| VEHCOUNT | 0.060225 |
| UNDERINFL | 0.042417 |
| ST_COLCODE | 0.040724 |
| SDOT_COLCODE | 0.038068 |
| LIGHTCOND | 0.036815 |

# ➢ <u>Results</u>

From the above study, we have created a working model based on the background of the problem which resulted in-

1. A Random Forest Model with an accuracy score of 74.05% built on 13 features
2. The three most important features of a car collision in Seattle are: Address Type, Pedestrians Count, and Collision Type

# ➢ <u>Discussion</u>

- Random Forest Model

As car collisions is a social topic, it is very hard to find a model that yields higher accuracy, as it is tough for a model to predict the outcome based on the limited attributes present in the study. So, 74% is reasonable very good accuracy of a learning model to predict the severity of a car collision. Further study & research of the factor's affecting the severity of an incident will help us in predicting the severity at a higher rate.

- Feature Importance

It is quite intuitive that, the ADDRTYPE is affecting the severity of a collision, as the address is a block, intersection, or alley can be very helpful in understanding the severity as they are guided by speed limits, the busyness of the area, and the probable angle of the collision. Also, the PEDCOUNT is a good indicator of the number of pedestrians involved in the collision as they are the most vulnerable users of the streets. The third and last most important factor is the Collision type itself. A head-on collision is more prone to severe damage rather than hitting a parked car. Hence, it is also a good indicator of the severity of the collision.

## ➢ **Conclusion**

The findings of the project can serve as a very good start for the institutions like Traffic Department & Emergency Health Care Service Providers. For example, based on the Address Type & Pedestrians injured, & Collision Type, the Health Care service Providers can make a firsthand estimate the severity of the collision as they explain 40% variance. Also, the traffic department can utilize this model to start predicting the severity of an accident.

## ***Phase 6: Deployment:***

The model is presented as a Jupyter notebook and the presentation as a ppt.