Capstone Project - Car accident severity

Introduction
A description of the problem and a discussion of the background.

There are a number of different factors that cause road collisions, in most cases it is related to driver factors, road and weather conditions. Collisions can result in terrible consequences, including death, injury, disability, property damage and financial costs.

For this project I will attempt to build a model which can predict the severity of an accident given the weather and the road conditions. Such a model could bring a new awareness to drivers of the dangers that can occur while travelling on the road and would encourage them to drive more safely.

The question we will attempt to answer is if you knew the weather and road conditions how severe would an accident be if a collision occurs?

Data
A description of the data and how it will be used to solve the problem.

The data we will use is provided by the SDOT Traffic Management Division and contains data of all types of collisions that happened in Seattle from 2004 to May/2020.

The data contains 194,673 samples and has 37 features that covers the weather and road conditions, collision factors and fatalities.

I will examine this data in detail in order to attempt to answer the question. I will do this by preparing the data to make the dataset readable and then apply 3 classification models on it. I will then discuss the results and apply conclusions for the report.

There is a lot of missing values in the data frame. Due to this we will not consider features with missing data. The columns in the data set which we are most interested in are:

- COLLISIONTYPE: Collision type
- WEATHER: Weather conditions during the time of the collision.
- ROADCOND: The condition of the road during the collision.
- LIGHTCOND: The light conditions during the collision.
- UNDERINFL: Whether or not a driver involved was under the influence of drugs or alcohol.

These columns do contain some missing values but it is below 3% of the total amount of samples. The target variable is SEVERITYCODE, this identifies the severity of the collision

- 1: Property Damage
- 2: Injury collision

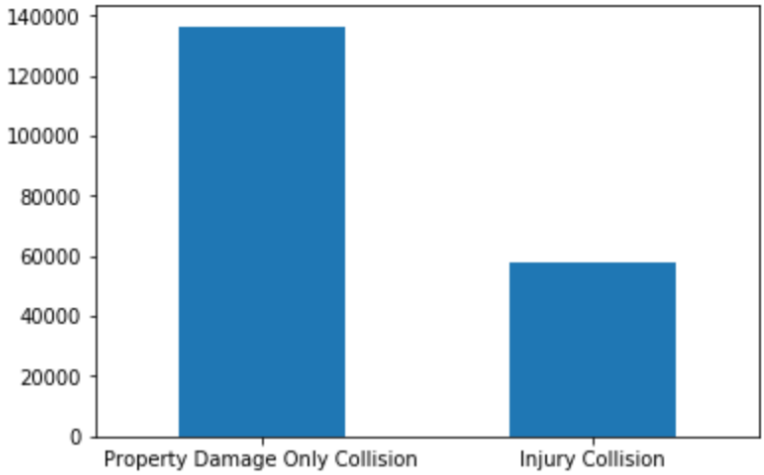
```
In [32]: df['SEVERITYDESC'].value_counts().to_frame()
```

Out[32]:

	SEVERITYDESC
Property Damage Only Collision	136485
Injury Collision	58188

```
In [33]: df['SEVERITYDESC'].value_counts().plot(kind='bar')
         plt.xticks(rotation=0)
```

```
Out[33]: (array([0, 1]), <a list of 2 Text xticklabel objects>)
```



Number of Annual Collisions

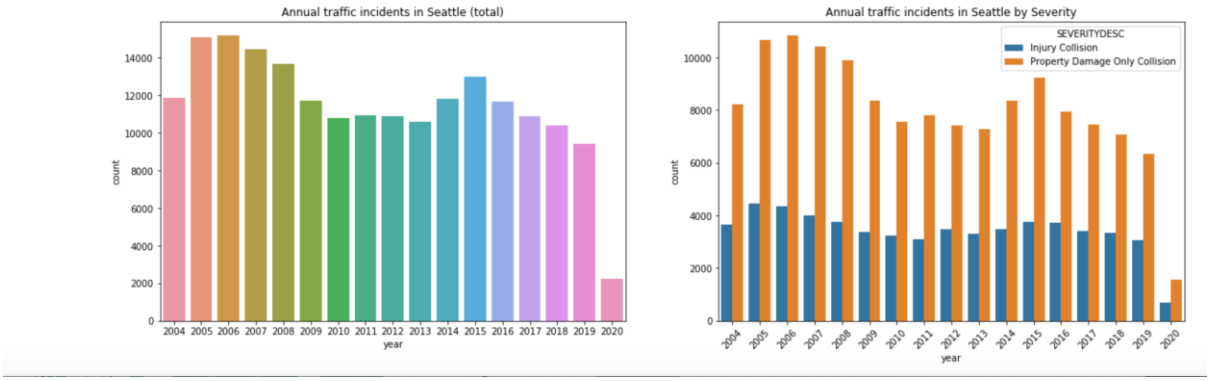
Note that the number for 2020 is lower as the data covers up to May 2020

Number of collisions year by year

```
In [34]:
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(20, 6))

df['year'] = pd.DatetimeIndex(df['INCDATE']).year
df['year'].value_counts().sort_index()#.plot(kind='bar')
sns.countplot(x="year", data=df, ax=ax1)
sns.countplot(x="year", hue="SEVERITYDESC", data=df, ax=ax2)
plt.xticks(rotation=45)
ax1.set_title('Annual traffic incidents in Seattle (total)')
ax2.set_title('Annual traffic incidents in Seattle by Severity')
```

```
Out[34]: Text(0.5, 1.0, 'Annual traffic incidents in Seattle by Severity')
```



Methodology

Data preparation and cleaning

We will clean the data to make the dataset more readable and suitable for the machine learning algorithms.

Removing Data

Of the 37 attributes, we will not consider the features with over 40% missing data, or other unclear and irrelevant data. We will use the COLLISIONTYPE, WEATHER, ROADCOND, LIGHTCOND and UNDERINFL data as attributes to classify the SEVERITYCODE. In order to do that we need to ensure the data is suitable for a binary classification model. We will use some popular machine learning algorithms to build up models and analyse their performance and predict the collision severity.

Working with missing values

The chosen attributes still has about 3% of data missing so we'll just drop them as there is still enough data to use.

Treating the categorical variables

As all attributes are categorical we will apply a label encoding technique on them.

Train/Test split and data normalization

In order to test/train the data the independent variables will be split into dataset X and the dependent variables 'SEVERITYCODE' to dataset Y.

We will now randomly pick samples and split in the radio of 70% to train the model and 30% to test the model.
After this split the data will be normalized to ensure the features are on a similar scale.

Classification: Modeling
We will use the dataset on 3 classification models:

- KNN: Classifies unseen data through the majority of its 'neighbours'. In this case we already know K=2 (2 classes of SEVERITY CODES). After obtaining each model's predictions we will evaluate their accuracy, precison, f1-score, log-loss and compare and discuss the results.

KNN Model

```
In [25]: from sklearn.neighbors import KNeighborsClassifier
         model_knn = KNeighborsClassifier(n_neighbors = 2).fit(X_train, y_train)
         model_knn

Out[25]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                              weights='uniform')
```

- Logistic Regression: Classifies data by estimating the probability of classes.

Logistic Regression

```
In [26]: from sklearn.linear_model import LogisticRegression

         model_lr = LogisticRegression(C=0.0001, solver='liblinear')
         model_lr.fit(X_train, y_train)
         model_lr

Out[26]: LogisticRegression(C=0.0001, class_weight=None, dual=False,
                            fit_intercept=True, intercept_scaling=1, max_iter=100,
                            multi_class='warn', n_jobs=None, penalty='l2', random_state=None,
                            solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

- Decision Tree: Classifies by breaking down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

Decision Tree

In [27]:
```
from sklearn.tree import DecisionTreeClassifier

model_tree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
model_tree.fit(X_train, y_train)
model_tree
```

Out[27]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                       splitter='best')
```

Evaluation using the data set

In [31]:
```
report = pd.DataFrame(data=np.array([knn_report, tree_report, lr_report]),
                      columns=['Algorithm', 'Jaccard', 'F1-score', 'Precision'])
report
```

Out[31]:

	Algorithm	Jaccard	F1-score	Precision
0	KNN	0.74	0.67	0.74
1	Decision Tree	0.75	0.69	0.78
2	Logistic Regression	0.7	0.58	0.68

The Jaccard score is used to compare a set of predicted labels for a sample to the corresponding set of labels. Here we can see that for all 3 models it is above 70%
The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The Decision Tree model presents the best F1 score.
Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. Again the Decision Model presents the best score

In [32]:
```
from sklearn.metrics import confusion_matrix

print('KNN Confusion Matrix')
tn, fp, fn, tp = confusion_matrix(y_test, yhat_knn).ravel()
(tn, fp, fn, tp)
```

```
KNN Confusion Matrix
```

Out[32]: (38708, 999, 13974, 3114)

In [33]:
```
print('Decision Tree Confusion Matrix')
tn, fp, fn, tp = confusion_matrix(y_test, yhat_tree).ravel()
(tn, fp, fn, tp)
```

```
Decision Tree Confusion Matrix
```

Out[33]: (39170, 537, 13755, 3333)

In [34]:
```
print('Logistic Regression Confusion Matrix')
tn, fp, fn, tp = confusion_matrix(y_test, yhat_lr).ravel()
(tn, fp, fn, tp)
```

```
Logistic Regression Confusion Matrix
```

Out[34]: (39655, 52, 16999, 89)

The confusion matrixes show the number of samples which classified correctly. We can see a big difference when comparing false positives and true positives, while there is a smaller difference with true negatives and false negatives

Discussion

We evaluated the performance of 3 machine learning algorithms on the given dataset in order to answer the question of predicting the severity of an accident knowing the weather and road conditions. The results of this performance for the 3 models was close, there was not a big difference in each of them. The Decision Tree model proved to be have the highest score in each category, followed by the KNN and then Logistic Regression.

Conclusion

For this project 5 out of 37 features were used which proved to be an adequate amount to resolve the problem. However if more features were used then we may have gotten different results. Additional features may also extract further data which could improve our models.