Kinetic Eye

Warehouse Risk Model Analysis

Problem Prompt:

We are looking to build a warehouse risk model, but we don't have a lot of accidents or injuries to refer to yet. One thought is that distance travelled by workers and vehicles is a potential predictor of risk. Let's assume that our safety events serve as a good proxy for safety outcomes. How well does distance travelled predict risk? What would you have to do to know for sure?

Clarification:

How well does the odometry data predict safety outcomes? The events data is intended to be the "outcomes." Consider using date/time column as a merging column.

In the problem, risk is measured at the operation level, rather than at the driver level. Consider looking at the hourly or daily level of granularity to measure the number of safety events.

This notebook will explore and analyze distance traveled to understand how the feature will perform in predicting risk. This notebook will also show the data cleaning, manipulation and merging process, as well as model results, conclusion, challenges, and next steps.

Imports

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime, date
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

Importing csv data files as pandas dataframe

In [2]:
```
odometry = pd.read_csv("odometry.csv")
events = pd.read_csv("events.csv")
```

In [3]:
```
odometry.head()
```

Out[3]:

	_id	camera_id	video_name	class_name	traveled_distance	!
0	ObjectId("5ecb9dc250efdb0008054f36")	5	040234-040603.mp4	Forklift	9.880006	2
1	ObjectId("5ecb9dc250efdb0008054f37")	5	040234-040603.mp4	Person	6.088650	2
2	ObjectId("5ecb9dc250efdb0008054f38")	5	040234-040603.mp4	Person	18.249779	2
3	ObjectId("5ecb9dc250efdb0008054f39")	5	040234-040603.mp4	Person	8.291737	2
4	ObjectId("5ecb9dc250efdb0008054f3a")	5	040234-040603.mp4	Forklift	8.895892	2

In [4]:
```
events.head()
```

Out[4]:

	Unnamed: 0	Date	Hour	Min	Event Type	Unnamed: 5
0	NaN	2020-05-24	22.0	28	SocialDistancing	NaN
1	NaN	2020-05-24	21.0	43	SocialDistancing	NaN
2	NaN	2020-05-24	20.0	27	SocialDistancing	NaN
3	NaN	2020-05-24	19.0	9	SocialDistancing	NaN
4	NaN	2020-05-25	11.0	2	FaceMask	NaN

Exploratory Data Analysis

Data Cleaning

In [5]:
```
# checking for empty or null values
print("Any null values in odometry?:", odometry.isnull().values.any())
print("Any null values in events?:", events.isnull().values.any())
```

```
Any null values in odometry?: False
Any null values in events?: True
```

In [6]:
```
# removing null columns
events = events.drop(['Unnamed: 0', 'Unnamed: 5'], axis=1)
print("Unique event types: ", np.unique(events['Event Type'].values))
print("The date column type is: ", type(events['Date'].values[0]))
print("Any null values in events?:", events.isnull().values.any())
```

```
Unique event types:  ['Driver entering' 'FaceMask' 'Gloves' 'Handcart R
iding' 'Other damage'
 'SocialDistancing' 'Speeding' 'Vests']
The date column type is:  <class 'str'>
Any null values in events?: False
```

In [7]:
```
# checking columns types
print("The timestamp columns type is: ", type(odometry['start_timestamp'
].values[0]))
print("The traveled_distance column type is: ", type(odometry['traveled_
distance'].values[0]))
```

```
The timestamp columns type is:  <class 'str'>
The traveled_distance column type is:  <class 'numpy.float64'>
```

In [8]:
```
# converting timestamp columns from str to datetime
odometry['start_timestamp'] = [datetime.strptime(time, "%Y-%m-%dT%H:%M:%
S.%fZ") for time in odometry['start_timestamp'].values]
odometry['end_timestamp'] = [datetime.strptime(time, "%Y-%m-%dT%H:%M:%S.
%fZ") for time in odometry['end_timestamp'].values]
odometry['local_start_timestamp'] = [datetime.strptime(time, "%Y-%m-%dT%
H:%M:%S.%fZ") for time in odometry['local_start_timestamp'].values]

events['Date'] = [datetime.strptime(date, "%Y-%m-%d") for date in events
['Date'].values]
```

In [10]:
```
# adding day and hour columns
odometry['day'] = odometry['local_start_timestamp'].dt.day

events['Day'] = events['Date'].dt.day


odometry['start_hour'] = odometry['start_timestamp'].dt.hour
```

Data Visualization

Distance Traveled

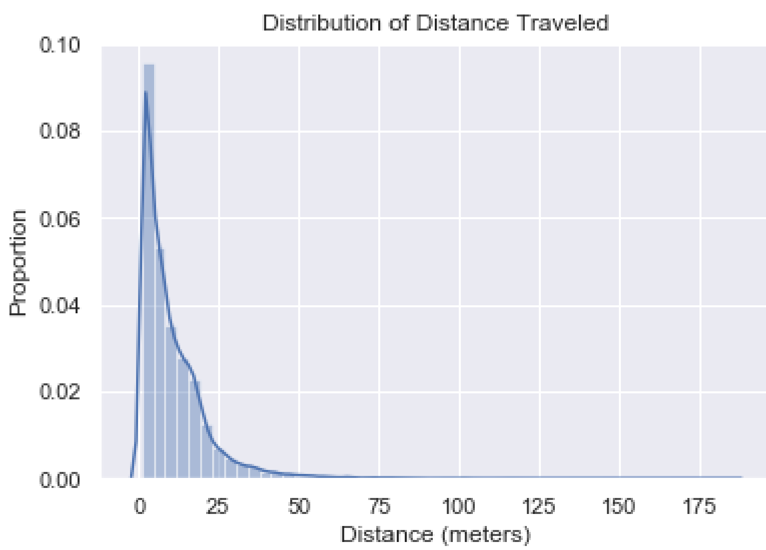
In [27]:
```
dist = odometry['traveled_distance'].values

sns.set(); np.random.seed(0)
ax = sns.distplot(dist)
ax.set_title('Distribution of Distance Traveled')
ax.set_xlabel("Distance (meters)")
ax.set_ylabel('Proportion')

print("Mean of Distance Traveled:", np.mean(dist))
print("Standard Deviation of Distance Traveled:", np.std(dist))
```

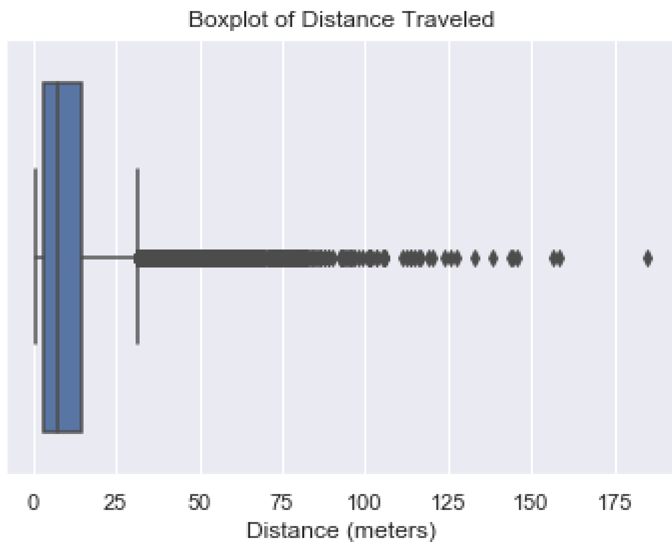
```
/Users/brianle/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.
py:1713: FutureWarning: Using a non-tuple sequence for multidimensional
indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In
the future this will be interpreted as an array index, `arr[np.array(se
q)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

Mean of Distance Traveled: 10.59894996265512
Standard Deviation of Distance Traveled: 10.818188997711168
```



In [28]:
```
ax = sns.boxplot(dist)
ax.set_title('Boxplot of Distance Traveled')
ax.set_xlabel("Distance (meters)")
```

Out[28]:
```
Text(0.5, 0, 'Distance (meters)')
```



Test: Log Transformation

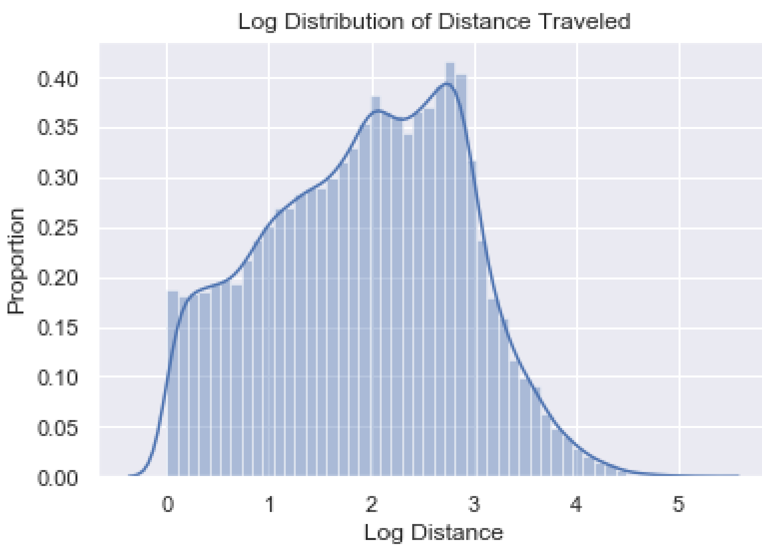
In [29]:
```
dist_log = np.log(dist)

ax = sns.distplot(dist_log)
ax.set_title('Log Distribution of Distance Traveled')
ax.set_xlabel("Log Distance")
ax.set_ylabel('Proportion')

print("Standard Deviation of Log Distance Traveled:", np.std(dist_log))
```

```
/Users/brianle/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.
py:1713: FutureWarning: Using a non-tuple sequence for multidimensional
indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In
the future this will be interpreted as an array index, `arr[np.array(se
q)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

Standard Deviation of Log Distance Traveled: 0.9629871196538377
```



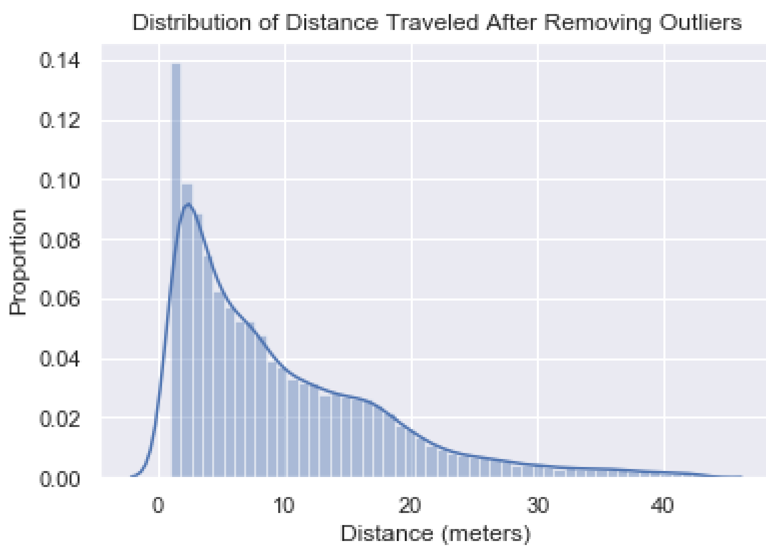
Test: Removing Outliers

In [30]:
```
dist_df = odometry['traveled_distance']
print("Number of trips:", len(dist_df))
dist_rmv_outliers = dist_df[(stats.zscore(dist_df)) < 3]
print("Number of trips after removing outliers:", len(dist_rmv_outliers
))
print("Proportion of trips left:", len(dist_rmv_outliers) / len(dist_df
))

ax = sns.distplot(dist_rmv_outliers.values)
ax.set_title('Distribution of Distance Traveled After Removing Outliers'
)
ax.set_xlabel("Distance (meters)")
ax.set_ylabel('Proportion')
```

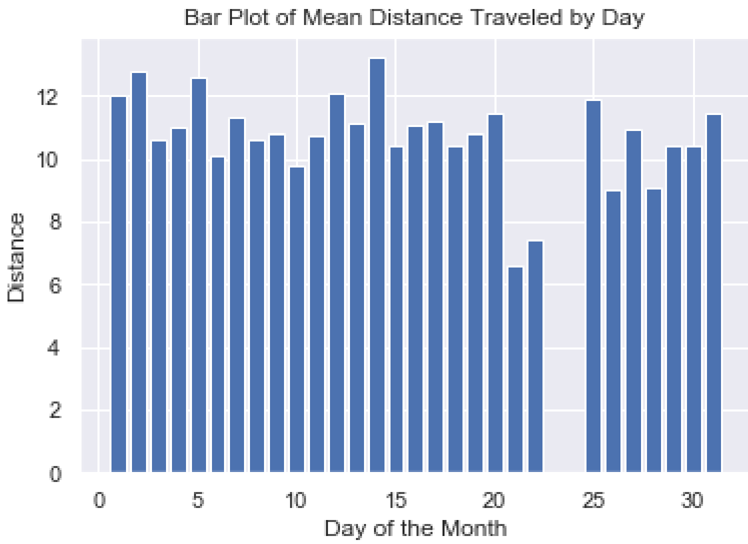
```
Number of trips: 40559
Number of trips after removing outliers: 39801
Proportion of trips left: 0.981311176311053
```

Out[30]:
```
Text(0, 0.5, 'Proportion')
```



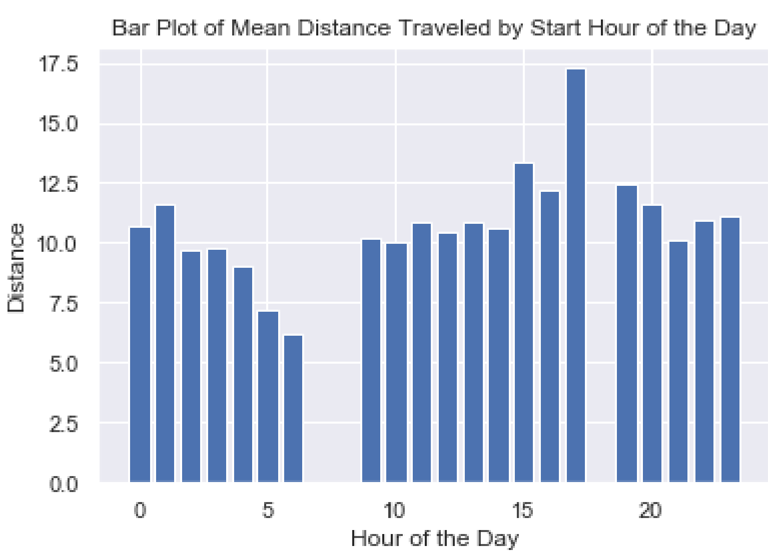
In [16]:
```
odo_by_day = odometry.groupby('day').agg(np.mean)
plt.bar(odo_by_day.index, odo_by_day.traveled_distance)
plt.xlabel('Day of the Month')
plt.ylabel('Distance')
plt.title('Bar Plot of Mean Distance Traveled by Day')
```

Out[16]:
```
Text(0.5, 1.0, 'Bar Plot of Mean Distance Traveled by Day')
```



In [17]:
```
odo_by_hour = odometry.groupby('start_hour').agg(np.mean)
plt.bar(odo_by_hour.index, odo_by_hour.traveled_distance)
plt.xlabel('Hour of the Day')
plt.ylabel('Distance')
plt.title('Bar Plot of Mean Distance Traveled by Start Hour of the Day')
```

Out[17]:
```
Text(0.5, 1.0, 'Bar Plot of Mean Distance Traveled by Start Hour of the
Day')
```



Distance Traveled Feature Analysis

In the first distribution plot, we find the data to be significantly right-skewed. Additionally, the boxplot shows a large number of outliers. In a regression model, skewed-data and outliers will have an adverse effect on model performance. To increase model performance, we must reduce the effect of outliers and skewness. Two ways of doing so is to either log transform the values or remove the outliers. After transforming the data or removing outliers, we find the distribution to be much more suitable for our model because there is less variablity. One thing to note is that removing outliers is not always the best method because we lose data, but in our case, we only lost about 2% of our data, so it is a viable method.

Moving forward, we will be removing outliers before inputting the data in our model.

Looking at the bar plot of mean distance traveled by day, we see that the data is missing values for the 23rd and 24th day. This could be due to a lack of data collection during these days. We will be filling these days with the average distance traveled among all the other mean distance traveled. In terms of the granularity of operation level, we will be measuring risk at the daily level because there is less variability than at the hourly level, which will ultimately provide better accuracy in our model.

Data Manipulation

In [31]:

```
odo_df = odometry.loc[:, ['traveled_distance', 'day']]

# removing outliers
odo_df = odo_df[(stats.zscore(odo_df['traveled_distance'])) < 3]

# grouping by day and aggregating by the mean
odo_by_day = odo_df.groupby('day').agg(np.mean)

# filling in missing values
mean_dist_trav = np.mean(odo_by_day['traveled_distance'])
odo_by_day['day'] = odo_by_day.index
days = np.append(odo_by_day['day'].values, [[23, 24]])
dists = np.append(odo_by_day['traveled_distance'], [[mean_dist_trav, mea
n_dist_trav]])
new_odo_day = pd.DataFrame({'Day' : days, 'mean_traveled_dist' : dists})
new_odo_day.head()
```

Out[31]:

	Day	mean_traveled_dist
0	1	10.435521
1	2	12.129680
2	3	9.470548
3	4	9.706520
4	5	10.842583

In [19]:
```
# grouping events by day, aggregating into a list and then summing up th
e number of safety outcomes for each day
grouped_events = events.groupby('Day').agg(list)
grouped_events['num_outcomes'] = [len(x) for x in grouped_events['Event
 Type']]
grouped_events.reset_index(inplace = True)
grouped_events = grouped_events.loc[:, ['Day', 'num_outcomes']]
grouped_events.head()
```

Out[19]:

	Day	num_outcomes
0	1	20
1	2	19
2	3	19
3	4	31
4	5	22

Merging

In [20]:
```
# merge on day of the month
merged_odo_events = new_odo_day.merge(grouped_events, left_on = 'Day', r
ight_on = 'Day')
merged_odo_events.head()
```

Out[20]:

	Day	mean_traveled_dist	num_outcomes
0	1	10.435521	20
1	2	12.129680	19
2	3	9.470548	19
3	4	9.706520	31
4	5	10.842583	22

Modeling

We will be using a linear regression model because distance traveled is our dependent variable and the number of outcomes is our independent variable.

In [33]:

```
X = merged_odo_events['mean_traveled_dist'].values.reshape(-1,1)
y = merged_odo_events['num_outcomes'].values.reshape(-1,1)

# splitting train and test set 80/20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2
0, random_state = 0)

regressor = LinearRegression().fit(X_train, y_train)
print('Intercept value:', regressor.intercept_[0])
print('Slope value:', regressor.coef_[0][0])
y_pred = regressor.predict(X_test)

model_df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred
.flatten()})
model_df.head()
```

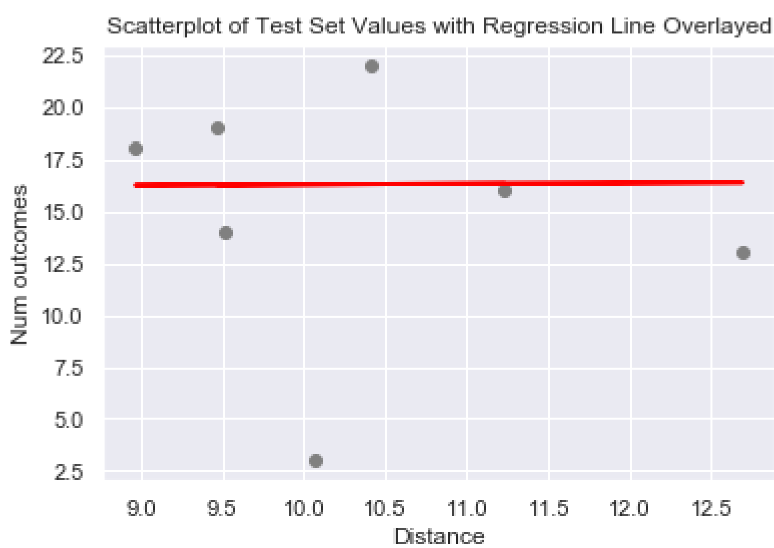
```
Intercept value: 15.946642721115522
Slope value: 0.03450682211795225
```

Out[33]:

	Actual	Predicted
0	19	16.273441
1	3	16.294314
2	13	16.384592
3	22	16.305925
4	14	16.275177

In [34]:

```
plt.scatter(X_test, y_test, color='gray')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.xlabel('Distance')
plt.ylabel('Num outcomes')
plt.title('Scatterplot of Test Set Values with Regression Line Overlaye
d')
plt.show()
```



In [23]:
```
print('Proportion of Variability in Y explained by X:', regressor.score(
X_test, y_test))
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred
))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_t
est, y_pred)))
```

```
Proportion of Variability in Y explained by X: -0.05526948130299214
Mean Absolute Error: 4.207557743558532
Mean Squared Error: 33.76862340169575
Root Mean Squared Error: 5.811077645471255
```

Conclusion

Based off the model results, we can conclude that the distance traveled does not perform well in our linear model to predict risk at the daily operational level, which is defined as the number of safety outcomes that occur in a day. The slope of the model and correlation coefficient is close to 0, meaning distance traveled has little to no effect on the number of safety outcomes. The MSE value of 33 shows that there is large variability within the residuals and that the model does not predict safety outcomes accurately, even after removing outliers from the input data.

Challenges

The granularity of operation level may be difficult to model due to lack of data points after aggregating. Using day of the month only provides at most 31 data points. If we were to use hourly granularity, there would be at most 24 data points assuming that warehouses operate up to 24 hours a day. I also attempted to groupby every single date in our dataset, but model performance did not improve.

Next Steps

To better predict risk, more data must be collected with different features that could serve as model inputs. We could also feature engineer new features and/or redefine our label and collect data accordingly.

In [ ]: