Problem Statement

Building digital maps is challenging, and maintaining it up to date in an ever-changing world is even more challenging. Various machine learning techniques helps us to detect road signs and changes in real world, and process it to update maps.

The problem presented here is related to a step after detecting a sign on a road. This step has to now identify each road geometry on which this sign is applicable. While sounds like a simple problem, signs in junctions makes this more challenging.

For example, given a sign detected on a road from a 4-camera setting on vehicle, the closest sighting of the sign may be in the right facing camera, with a sharp sign angle with respect to the direction of the car on which cameras set is mounted. Next step for updating map using this sign is to identify the exact road on which this sign is to be placed or applied.

On a + junction, when a sign is detected on the right camera, its hard now to tell if this sign is for the straight road, or for the right-side road, unless you consider parameters like sign bounding box aspect ratio.

For example, a sign detected from Front camera will have a natural aspect ratio of the sign when it is actually facing front of the car, however when same sign is detected on a right-side camera with a sharp angle from front, sign bounding box gets skewed, giving a hint that although its detected in right, it’s still facing the front of the car.

Dataset provided here has details on camera sign was detected, Angle of sign with respect to front in degrees, Sign's reported bounding box aspect ratio (width/height), Sign Width and Height, and the target feature Sign Facing, which is where the sign is actually facing.

Goal here is to predict where the sign is actually facing with respect to the vehicle, given above set of inputs

In [1]:

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier

train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

In [2]:

```
train.head()
```

Out[2]:

	Id	DetectedCamera	AngleOfSign	SignAspectRatic
0	2c9180975a056a64015a1e0a52e57021	Rear	195	1.02
1	2c9180975a056a64015a1e17b32171e4	Rear	203	1.09
2	2c9180975a056a64015a1de4deb16bd5	Front	26	0.96
3	2c9180975a056a64015a1de4deb16bdd	Rear	199	0.81
4	2c9180975a056a64015a1de4deb16bd6	Rear	208	0.93

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

Out[3]:

	Id	DetectedCamera	AngleOfSign	SignAspectRatio
0	2c9180975a056a64015a1e10d3f270fe	Right	67	0.63
1	2c9180975a056a64015a1de4deb16bdc	Front	16	0.88
2	2c9180975a056a64015a1e0e70ea70ce	Right	44	1.15
3	2c9180975a056a64015a1dfed0c46ec6	Right	50	1.10
4	2c9180975a056a64015a1dfed0c46ec7	Front	30	0.95

In [4]:
```
train['DetectedCamera'].value_counts()
```

Out[4]:
```
Front    10910
Right    10516
Left      9298
Rear      7761
Name: DetectedCamera, dtype: int64
```

In [5]:
```
#encode as integer
mapping = {'Front':0, 'Right':1, 'Left':2, 'Rear':3}
train = train.replace({'DetectedCamera':mapping})
test = test.replace({'DetectedCamera':mapping})
```

In [6]:
```
#renaming column
train.rename(columns = {'SignFacing (Target)': 'Target'}, inplace=True)
```

In [7]:
```
#encode Target Variable based on sample submission file
mapping = {'Front':0, 'Left':1, 'Rear':2, 'Right':3}
train = train.replace({'Target':mapping})
```

In [8]:
```
#target variable
y_train = train['Target']
test_id = test['Id']
```

In [9]:
```
#drop columns
train.drop(['Target','Id'], inplace=True, axis=1)
test.drop('Id',inplace=True,axis=1)
```

In [28]:

```
#train model
#clf = RandomForestClassifier(n_estimators=500,max_features=3,min_samples_spli
t=5,oob_score=True)
clf=RandomForestClassifier(n_estimators = 1200, oob_score=True, n_jobs =-1,ran
dom_state =42,
                           max_features ="auto", min_samples_leaf=1, min_sampl
es_split=50)
clf.fit(train, y_train)
```

Out[28]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_samples_leaf=1, min_samples_split=50,
            min_weight_fraction_leaf=0.0, n_estimators=1200, n_jobs=-1,
            oob_score=True, random_state=42, verbose=0, warm_start=False)
```

In [ ]:

```
#predict on test data
from sklearn.svm import SVC
model =SVC()
model.fit(train,y_train)
pred = model.predict_proba(test)
```

In [37]:

```
#write submission file and submit
columns = ['Front','Left','Rear','Right']
sub = pd.DataFrame(data=pred, columns=columns)
sub['Id'] = test_id
sub = sub[['Id','Front','Left','Rear','Right']]
sub.to_csv("sub2.csv",index=False, float_format='%0.6f')
```

In [10]:

```
from sklearn.svm import SVC
```

In [ ]:

```
# Support Vector Machines
svc = SVC(probability=True)
svc.fit(train, y_train)
Y_pred = svc.predict(test)
#acc_svc = round(svc.score(train, y_train) * 100, 2)
#acc_svc
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: