Level One Patterns code

August 10, 2020

[1]:
```
%matplotlib inline

from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split

import numpy as np
import pandas as pd

import pylab as pl
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches

import os

from PIL import Image

from ast import literal_eval

plt.rcParams['figure.figsize'] = [20, 10]

map_img = Image.open('lec.png')
map_img = map_img.resize((700, 700))
```

[2]:
```
df = pd.read_csv("g2redside.csv")

secs = df['Seconds'].tolist()
df.drop(['Unnamed: 0','Seconds'],
         inplace=True,
         axis=1)
cols = df.columns.tolist()
```

[3]:
```
X = np.empty([len(secs)*df.shape[1],3])
y = np.empty(len(secs)*df.shape[1])

k = 0
lowest = min(secs)

for i in secs:
    for y_j, j in enumerate(df.loc[i-lowest]):
        j = literal_eval(j.replace('(nan, nan)', '(0,0)'))
        X[k] = [i, j[0],j[1]]
        y[k] = y_j % 10
        k+=1
```

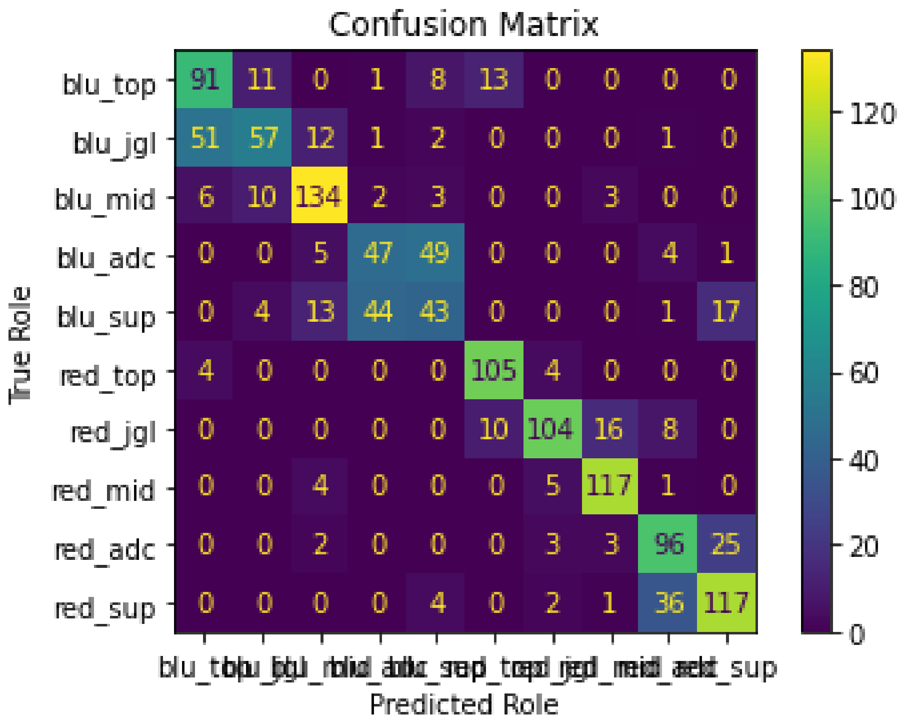
[4]:
```
y = (y[[0 not in i for i in X]])
X = (X[[0 not in i for i in X]])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,␣
 ↪random_state=0)
```

[5]:
```
clf = LogisticRegression(random_state=123,
                         multi_class='multinomial',
                         solver='newton-cg')
model = clf.fit(X_train, y_train)
```

[6]:
```
clf.score(X_test,y_test)
```

[6]:
```
0.7002305918524212
```

[7]:
```
metrics.plot_confusion_matrix(model, X_test,y_test,values_format='d')
plt.title("Confusion Matrix")
plt.xticks(np.
 ↪arange(10),labels=['blu_top','blu_jgl','blu_mid','blu_adc','blu_sup','red_top','red_jgl','r
plt.yticks(np.
 ↪arange(10),labels=['blu_top','blu_jgl','blu_mid','blu_adc','blu_sup','red_top','red_jgl','r
plt.xlabel("Predicted Role")
plt.ylabel("True Role")


plt.show()
```

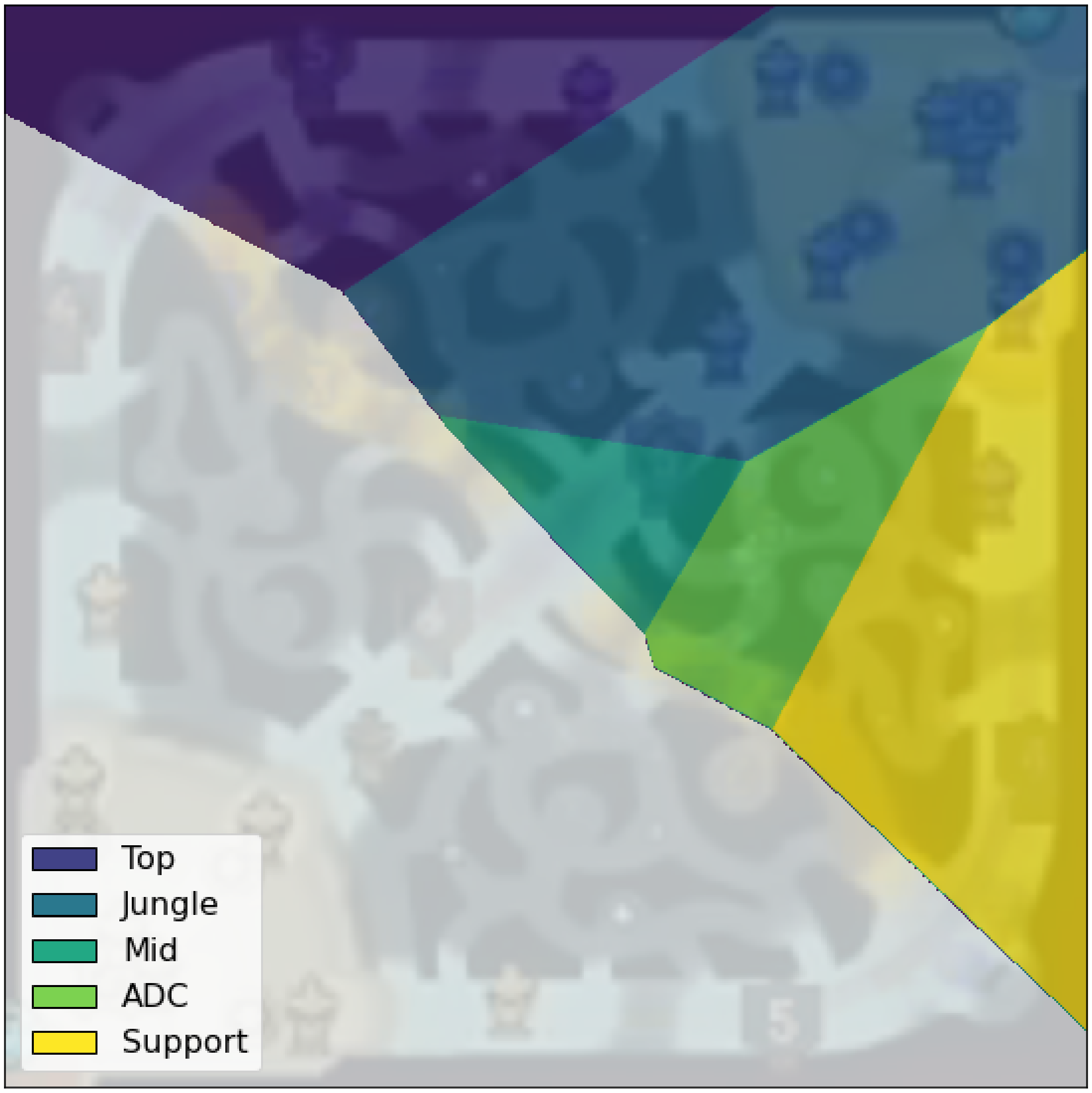


[8]:
```
x_mesh, y_mesh = np.meshgrid(np.linspace(0,150, 700),
                             np.linspace(0,149, 700))

timer = 60

time_mesh = np.zeros_like(x_mesh)
time_mesh[:,:] = timer

grid_predictor_vars = np.array([time_mesh.ravel(),
                                x_mesh.ravel(),
                                y_mesh.ravel()]).T
```

[9]:
```
preds = model.predict(grid_predictor_vars)
```

[10]:
```
top = cm.get_cmap('Blues', 128)
bottom = cm.get_cmap('viridis', 128)

newcolors = np.vstack((top(np.linspace(0, 0, 128)),
                       bottom(np.linspace(0, 1, 128))))
newcmp = ListedColormap(newcolors, name='OrangeBlue')

plt.figure(figsize=(20,10))
plt.imshow(preds.reshape(x_mesh.shape),
            cmap = newcmp,
            resample = True)
plt.xticks([])
plt.yticks([])
plt.title("Player most likely to be the one found at each spot at %d seconds" %␣
 ↪timer)
plt.imshow(map_img,
            zorder=1,
            alpha = 0.25,
            interpolation= "nearest")

top_patch = mpatches.Patch(color=newcmp(0.6), label='Top', ec="black")
jgl_patch = mpatches.Patch(color=newcmp(0.7), label='Jungle', ec="black")
mid_patch = mpatches.Patch(color=newcmp(0.8), label='Mid', ec="black")
adc_patch = mpatches.Patch(color=newcmp(0.9), label='ADC', ec="black")
sup_patch = mpatches.Patch(color=newcmp(1.0), label='Support', ec="black")


plt.legend(loc="lower left",handles=[top_patch,jgl_patch, mid_patch, adc_patch,␣
 ↪sup_patch], prop={'size': 16})#
plt.show()
```

Player most likely to be the one found at each spot at 60 seconds

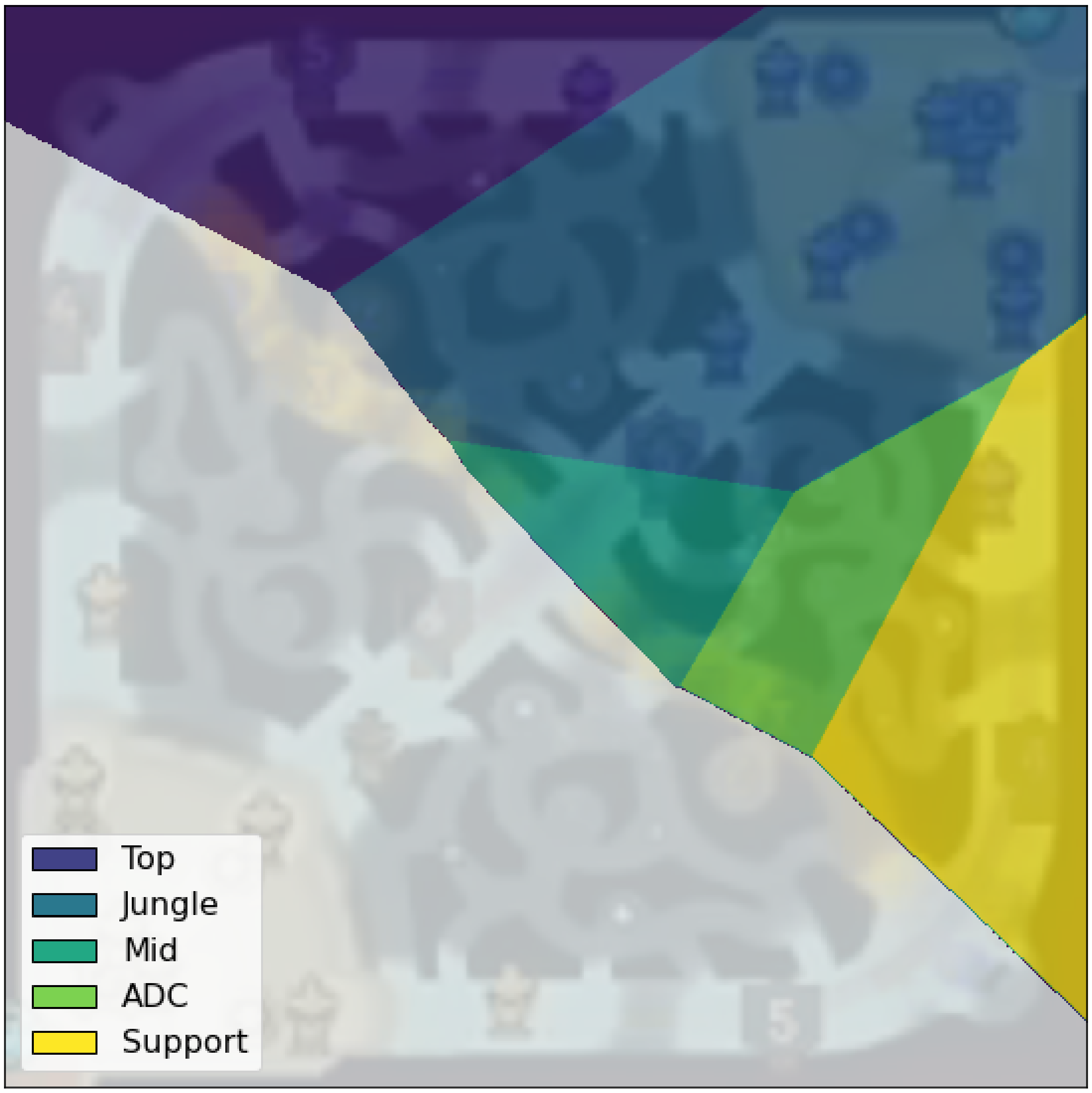


```
[11]: def red_levelonefull(timer):
          x_mesh, y_mesh = np.meshgrid(np.linspace(0,150, 700),
                                       np.linspace(0,149, 700))

          time_mesh = np.zeros_like(x_mesh)
          time_mesh[:,:] = timer

          grid_predictor_vars = np.array([time_mesh.ravel(),
                                          x_mesh.ravel(),
                                          y_mesh.ravel()]).T

          preds = model.predict(grid_predictor_vars)
```

```
    top = cm.get_cmap('Blues', 128)
    bottom = cm.get_cmap('viridis', 128)

    newcolors = np.vstack((top(np.linspace(0, 0, 128)),
                           bottom(np.linspace(0, 1, 128))))
    newcmp = ListedColormap(newcolors, name='OrangeBlue')

    plt.figure(figsize=(20,10))
    plt.imshow(preds.reshape(x_mesh.shape),
               cmap = newcmp,
               resample = True)
    plt.xticks([])
    plt.yticks([])
    plt.title("Player most likely to be the one found at each spot at %d␣
↪seconds" % timer)
    plt.imshow(map_img,
               zorder=1,
               alpha = 0.25,
               interpolation= "nearest")

    top_patch = mpatches.Patch(color=newcmp(0.6), label='Top', ec="black")
    jgl_patch = mpatches.Patch(color=newcmp(0.7), label='Jungle', ec="black")
    mid_patch = mpatches.Patch(color=newcmp(0.8), label='Mid', ec="black")
    adc_patch = mpatches.Patch(color=newcmp(0.9), label='ADC', ec="black")
    sup_patch = mpatches.Patch(color=newcmp(1.0), label='Support', ec="black")


    plt.legend(loc="lower left",handles=[top_patch,jgl_patch, mid_patch,␣
↪adc_patch, sup_patch], prop={'size': 16})
```

[12]:
```
red_levelonefull(80)
```

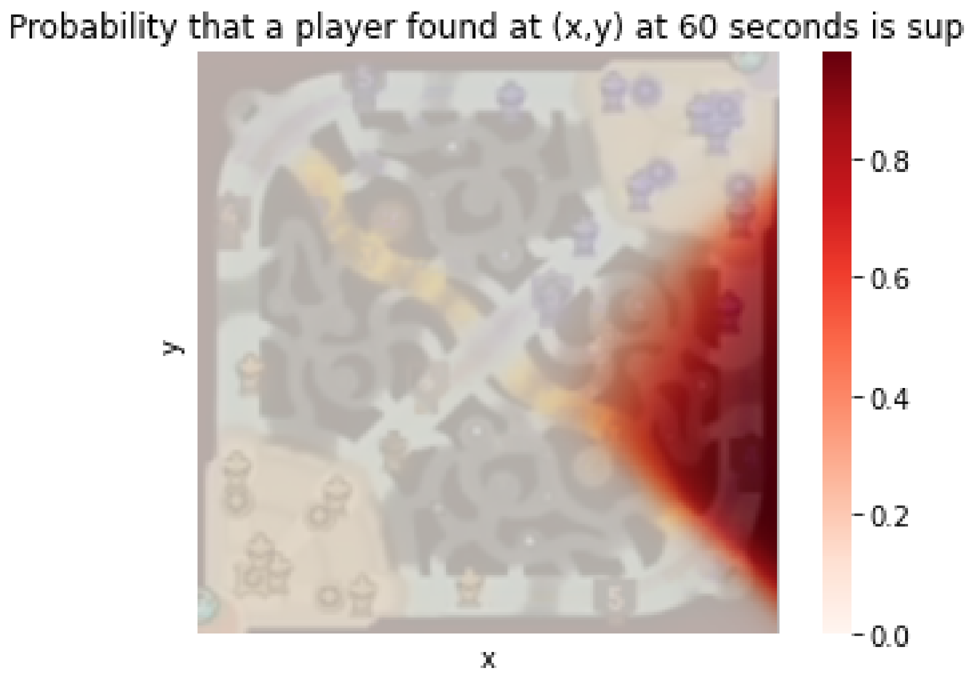
Player most likely to be the one found at each spot at 80 seconds

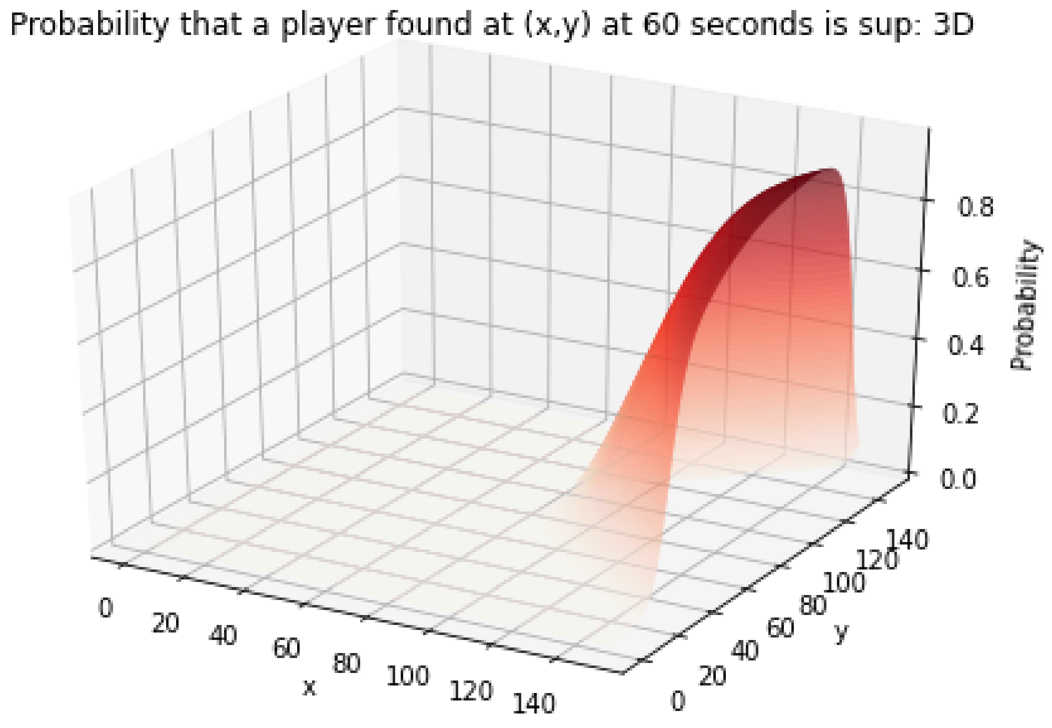


```
[13]: def red_leveloneplayer(timer, position):
          positions = dict(zip(['top','jgl','mid','adc','sup'], [5,6,7,8,9]))
          x_mesh, y_mesh = np.meshgrid(np.linspace(0,150, 700),
                                       np.linspace(0,149, 700))

          time_mesh = np.zeros_like(x_mesh)
          time_mesh[:,:] = timer

          grid_predictor_vars = np.array([time_mesh.ravel(),
                                          x_mesh.ravel(),
                                          y_mesh.ravel()]).T
          preds = model.predict_proba(grid_predictor_vars)
```

```
    hp_mesh = preds.T[positions[position]].reshape(x_mesh.shape)

    hm = sns.heatmap(hp_mesh, cmap="Reds",
                     yticklabels=False,
                     xticklabels=False)
    plt.title("Probability that a player found at (x,y) at %d seconds is %s" %␣
↪(timer,position))
    plt.xlabel('x')
    plt.ylabel('y')
    plt.imshow(map_img, zorder=1, alpha = 0.3)

    fig = pl.figure()
    ax = Axes3D(fig)

    fig = ax.plot_surface(x_mesh, y_mesh, hp_mesh,
        rstride=1, cstride=1, cmap='Reds',lw=0.1)
    ax.set_title("Probability that a player found at (x,y) at %d seconds is %s:␣
↪3D" % (timer,position))
    ax.set_ylabel("y")
    ax.set_xlabel("x")
    ax.set_zlabel("Probability")
    pl.show()
```

```
[14]: red_leveloneplayer(60, 'sup')
```

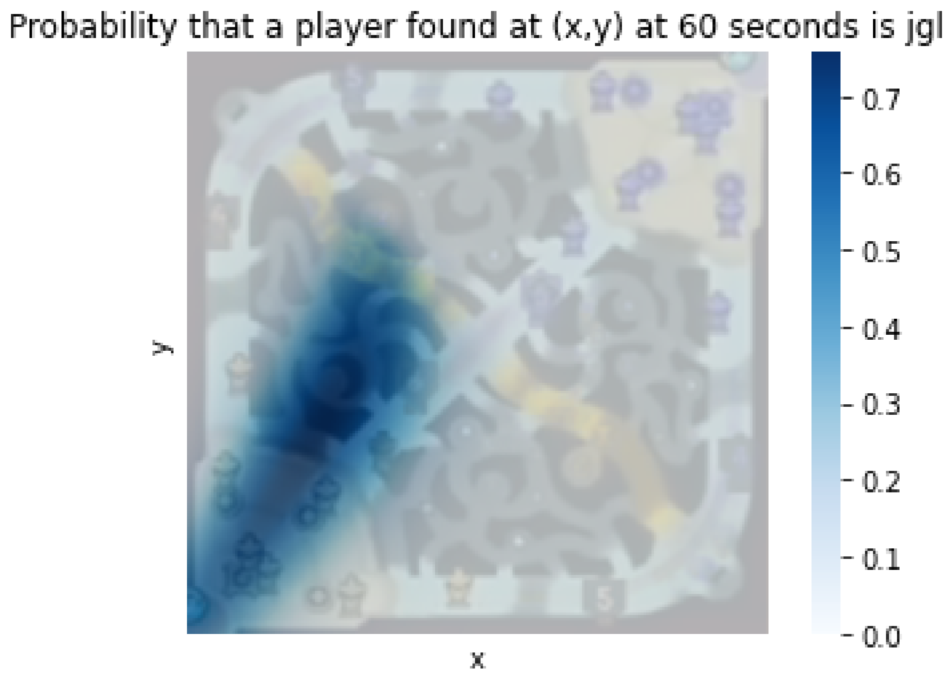


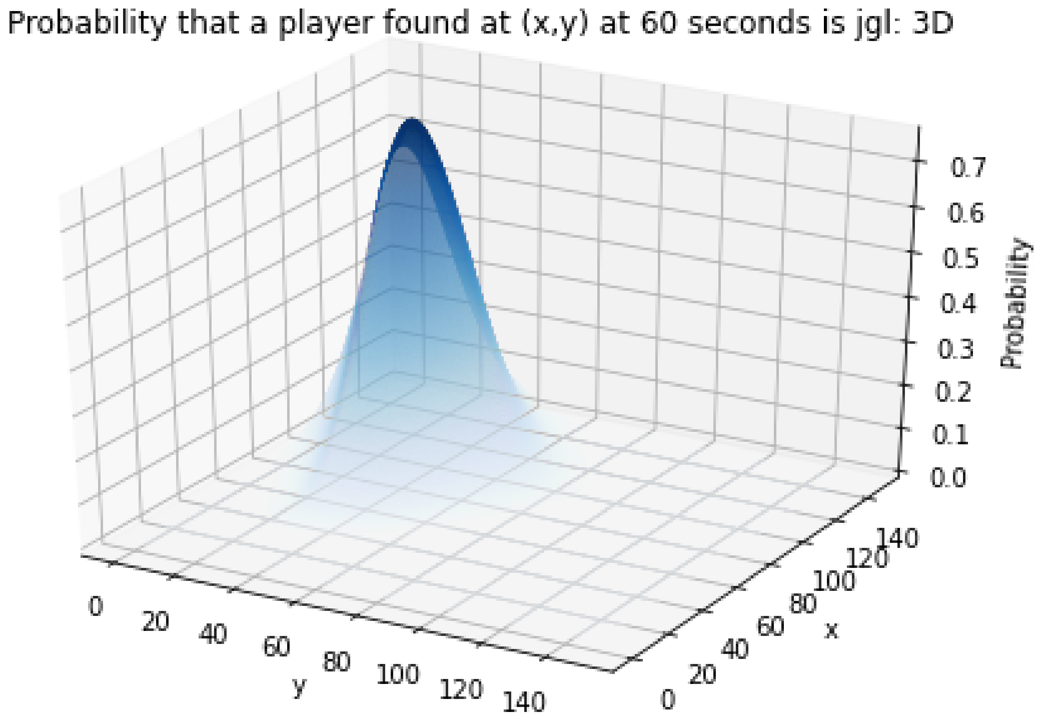


[15]:
```
blue_df = pd.read_csv("g2blueside.csv")

secs = blue_df['Seconds'].tolist()
blue_df.drop(['Unnamed: 0','Seconds'],
         inplace=True,
         axis=1)

X = np.empty([len(secs)*blue_df.shape[1],3])
y = np.empty(len(secs)*blue_df.shape[1])

k = 0
lowest = min(secs)

for i in secs:
    for y_j, j in enumerate(blue_df.loc[i-lowest]):
        j = literal_eval(j.replace('(nan, nan)', '(0,0)'))
        X[k] = [i, j[0],j[1]]
        y[k] = y_j % 10
        k+=1

cols = blue_df.columns.tolist()
```

```
y = (y[[0 not in i for i in X]])
X = (X[[0 not in i for i in X]])

clf = LogisticRegression(random_state=0,
                         multi_class='multinomial',
                         solver='newton-cg')
blue_model = clf.fit(X, y)

def blue_levelonefull(timer):
    x_mesh, y_mesh = np.meshgrid(np.linspace(0,150, 700),
                                 np.linspace(0,149, 700))

    time_mesh = np.zeros_like(x_mesh)
    time_mesh[:,:] = timer

    grid_predictor_vars = np.array([time_mesh.ravel(),
                                    x_mesh.ravel(),
                                    y_mesh.ravel()]).T

    preds = blue_model.predict(grid_predictor_vars)

    top = cm.get_cmap('viridis', 128)
    bottom = cm.get_cmap('Reds', 256)

    newcolors = np.vstack((top(np.linspace(0, 1, 128)),
                           bottom(np.linspace(0, 0, 128))))
    newcmp = ListedColormap(newcolors, name='OrangeBlue')

    plt.figure(figsize=(20,10))
    plt.imshow(preds.reshape(x_mesh.shape),
               cmap = newcmp,
               resample = True)
    plt.xticks([])
    plt.yticks([])
    plt.title("Player most likely to be the one found at each spot at %d␣
 ↪seconds" % timer)
    plt.imshow(map_img,
               zorder=1,
               alpha = 0.25,
               interpolation= "nearest")

    top_patch = mpatches.Patch(color=newcmp(0), label='Top', ec="black")
    jgl_patch = mpatches.Patch(color=newcmp(0.1), label='Jungle', ec="black")
    mid_patch = mpatches.Patch(color=newcmp(0.2), label='Mid', ec="black")
    adc_patch = mpatches.Patch(color=newcmp(0.3), label='ADC', ec="black")
    sup_patch = mpatches.Patch(color=newcmp(0.4), label='Support', ec="black")
    plt.legend(loc="upper right",handles=[top_patch,jgl_patch, mid_patch,␣
 ↪adc_patch, sup_patch], prop={'size': 16})


def blue_leveloneplayer(timer, position):
    positions = dict(zip(['top','jgl','mid','adc','sup'], [0,1,2,3,4]))
    x_mesh, y_mesh = np.meshgrid(np.linspace(0,150, 700),
                                 np.linspace(0,149, 700))

    time_mesh = np.zeros_like(x_mesh)
    time_mesh[:,:] = timer

    grid_predictor_vars = np.array([time_mesh.ravel(),
                                    x_mesh.ravel(),
                                    y_mesh.ravel()]).T
    preds = blue_model.predict_proba(grid_predictor_vars)

    hp_mesh = preds.T[positions[position]].reshape(x_mesh.shape)

    hm = sns.heatmap(hp_mesh, cmap="Blues",
                     yticklabels=False,
                     xticklabels=False)
    plt.title("Probability that a player found at (x,y) at %d seconds is %s" %␣
 ↪(timer,position))
    plt.xlabel('x')
    plt.ylabel('y')
    plt.imshow(map_img, zorder=1, alpha = 0.3)

    fig = pl.figure()
    ax = Axes3D(fig)

    fig = ax.plot_surface(x_mesh, y_mesh, hp_mesh,
        rstride=1, cstride=1, cmap='Blues',lw=0.01)
    ax.set_title("Probability that a player found at (x,y) at %d seconds is %s:␣
 ↪3D" % (timer,position))
    ax.set_ylabel("x")
    ax.set_xlabel("y")
    ax.set_zlabel("Probability")

    pl.show()
```

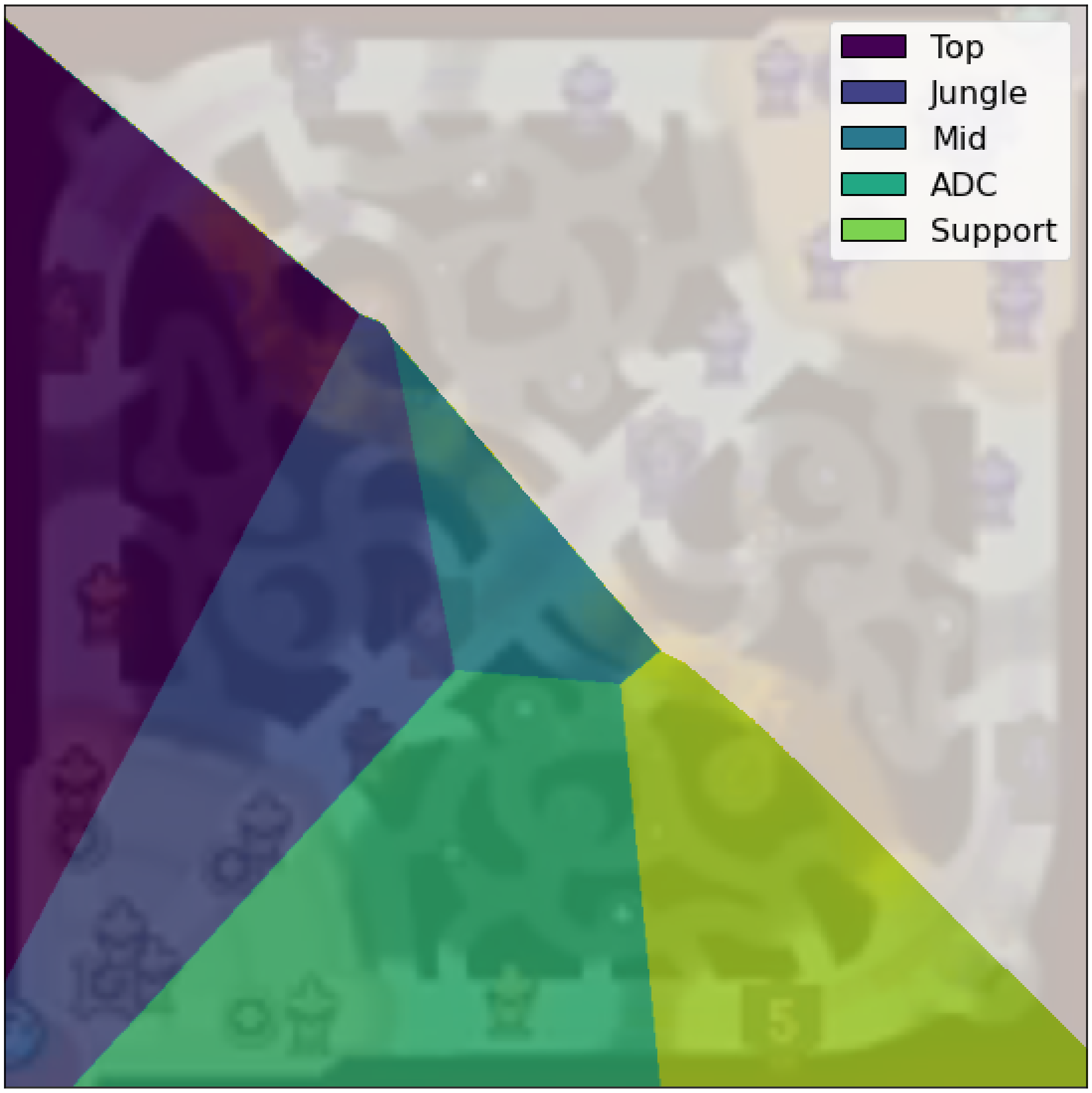
[16]:
```
blue_leveloneplayer(60,"jgl")
```


Probability that a player found at (x,y) at 60 seconds is jgl
y
x
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0


Probability that a player found at (x,y) at 60 seconds is jgl: 3D
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Probability
0
20
40
60
80
100
120
140
y
0
20
40
60
80
100
120
140
x

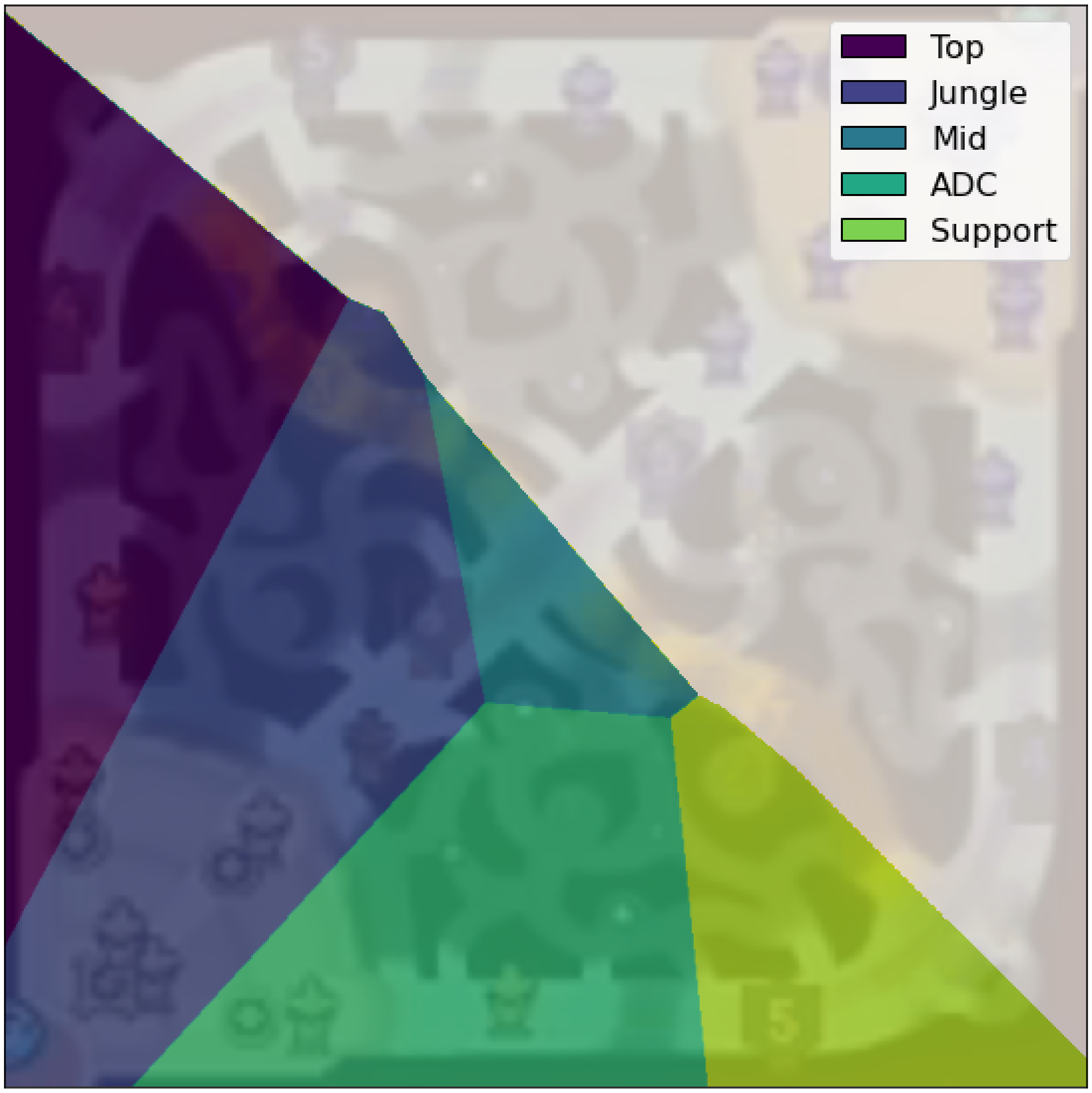
```
[17]: blue_levelonefull(60)
```

Player most likely to be the one found at each spot at 60 seconds



```
[18]: blue_levelonefull(80)
```

Player most likely to be the one found at each spot at 80 seconds


Top
Jungle
Mid
ADC
Support