Продвинутое задание 2. Яндекс такси

Задача: По данным о поездке определить точку посадки клиента в такси.

Методы решения:

1) Медиана

2) Точка, в которой машина провела больше всего времени.

Метрика для оценки качества предложенного решения:

Так как пользователю требуется примерно 20-30 секунд, чтобы сесть в машину, то оценка считается *успешной*, если в ее окрестности машина останавливалась хотя бы на 20 секунд. Оценкой качества точки посадки считается доля *успешных* предсказаний.

В данном решении погрешность GPS считается 2 метра.

In [65]:
```
import ast
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

%matplotlib inline
```

In [2]:
```
file = open('stop_points_data.txt', 'r')
data = file.readlines()
```

In [56]:
```
def median_estimator(string):
    """
    Принимает строку, запись в журнале;
    Возвращает медиану среди точек ожидания.
    """
    session = pd.DataFrame(ast.literal_eval(string[8:-1]))
    # в оценке участвуют только точки ожидания и первые точки поездки с клиентом
    # начало поездки учитывается потому что многие водители сначала сменяют статус, а потом только начинают ехать.
    session = pd.concat([session[session.status==1], session[session.status==2].head(2)])
    return (session.x.median(), session.y.median())
median_estimator = np.vectorize(median_estimator)
```

In [58]:
```
def longest_stay_estimator(string, accuracy=2):
    """
    Принимает строку, запись в журнале;
    Возвращает оценку точки посадки, находя точку, в которой машина находилась дольше всего.
    """
    session = pd.DataFrame(ast.literal_eval(string[8:-1]))
    # в оценке участвуют только точки ожидания и первые точки поездки с клиентом
    # начало поездки учитывается потому что многие водители сначала сменяют статус, а потом только начинают ехать.
    session = pd.concat([session[session.status==1], session[session.status==2].head(2)])

    # координаты округляются, чтобы уменьшить погрешность GPS. Т.е соседние точки теперь имеют одну координату
    session['x_round'] = np.round(session.x/accuracy)*accuracy
    session['y_round'] = np.round(session.y/accuracy)*accuracy

    # находится координата, в которую попало больше всего точек
    areas = session.groupby(['x_round', 'y_round'])['x', 'y'].agg(['count','mean'])
    stop = areas.x['count'].idxmax()

    # возвращается среднее среди соседних координат, считавшихся за одну
    return (areas.x['mean'][stop], areas.y['mean'][stop])

longest_stay_estimator= np.vectorize(longest_stay_estimator)
```

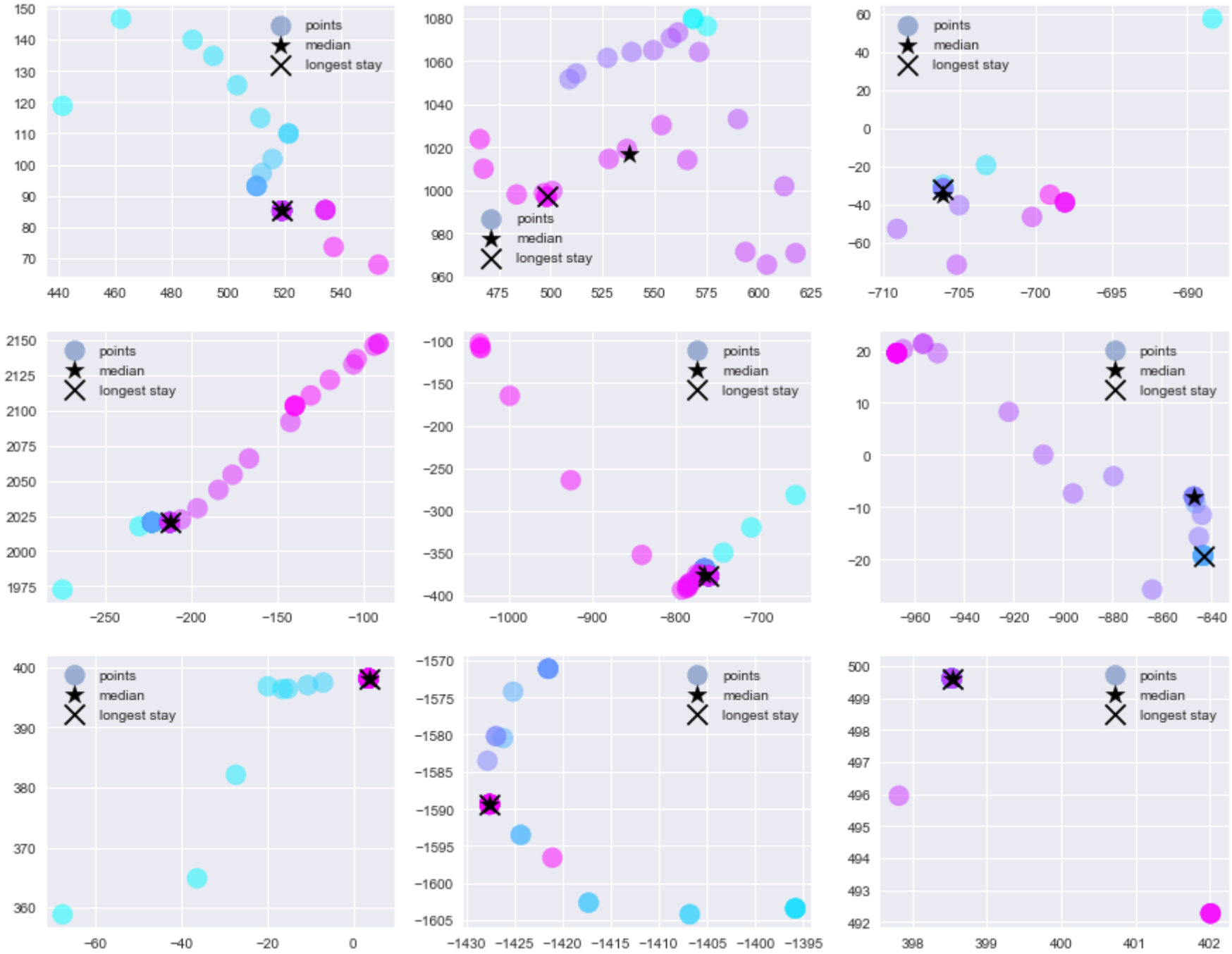
Нарисуем маршруты во время ожидания и точки остановки.

In [66]:

```
plt.figure(figsize=(15, 12))
for i in range(9):
    plt.subplot(3, 3, i+1)

    session = pd.DataFrame(ast.literal_eval(data[i][8:-1]))
    session = pd.concat([session[session.status==0].tail(2),
                         session[session.status==1],
                         session[session.status==2].head(2)])

    #ранние точки имеют голубой цвет, поздние розовый
    plt.scatter(session.x, session.y, c=session.ts, cmap='cool', s=200, alpha=0.5, label='points')
    estimator = median_estimator(data[i])
    plt.scatter(estimator[0], estimator[1], marker='*', s=200, color='black', label='median')
    estimator = estimate_stop(data[i])
    plt.scatter(estimator[0], estimator[1], marker='x', s=200, color='black', label='longest stay')
    plt.legend()
plt.show()
```


points
median
longest stay

Можно увидеть, что иногда методы дают разные оценки.

Следующие функции позволяют оценить качество метода оценки. Как уже говорилось, оценка качества состоит подсчете доли удачных предсказаний.

In [60]:
```
def dist(p1, p2):
    "Расстояние между точками"
    return ((p1[0]-p2[0])**2 + (p1[1]-p2[1])**2)**(1/2)
```

In [61]:
```
def time_spend_at_point(string, point, accuracy=2):
    """
    Функция оценивает качество предложеного для данной сессии решения.

    Принимает:
    string - запись сессии такиста
    point - оцениваемая точка
    accuracy - радиус окрестности

    Возвращает время, которое машина находилась в окрестности точки point.
    """
    session = pd.DataFrame(ast.literal_eval(string[8:-1]))
    session = pd.concat([session[session.status==0].tail(2),
                         session[session.status==1],
                         session[session.status==2].head(2)])
    # считается наибольший промежуток, во время которого машина находилась в трехметровой окрестности точки.
    start = session.ts.min()
    stop = 0
    max_time = 0
    for i, row in session.iterrows():
        if dist((row.x, row.y), point) < accuracy:
            stop = row.ts
        else:
            start = row.ts
        max_time = max(max_time, stop - start)
    return max_time
```

In [62]:
```
def evaluate_estimator(data, estimators, accuracy=2, min_stay_time=20):
    """Возвращает долю успеха"""
    success = 0
    for i in range(len(data)):
        if time_spend_at_point(data[i], (estimators[0][i], estimators[1][i]), accuracy) > min_stay_time:
            success += 1
    return success/len(data)
```

In [63]:
```
sample = data[:1000]
evaluate_estimator(sample, median_estimator(sample))
```

Out[63]:
```
0.615
```

In [64]:
```
evaluate_estimator(sample, longest_stay_estimator(sample))
```

Out[64]:
```
0.813
```

Итак, доля успешных предсказаний для медианы равна 0.6, а для второго метода 0.8. Значит, второй метод дает лучшие предсказания, однако он работает дольше.

In [ ]:
```

```