1) Чистим и Исследуем 2) Преобразуем 3) Моделируем 4) Ансамблирование

Чистим и Исследуем

train.describe(), train.info(), train.nunique(),
статистическая информация по числовым, информация по всем столбцам

train['col_name'].unique(),
массив уникальных значений в 'col_name', только для Series; numpy.ndarray

train['col_name'].value_counts(sort=True, dropna=False)
каждому уникальному значению из столбца 'col_name' ее количество; Series

train[train['col'].isnull()], train[train['col'].isna()],
вывод строк с пустыми элем-ми в столбце 'col'

train.columns,
вывод названий столбцов, можно использовать для присвоения

train.loc([100500],['col_name'])
вывод 100500-го элемента, можно использовать для присвоения

import seaborn as sns
распространенный пакет визуализации данных.

sns.regplot(x='Age', y='Fare', data = test)
линейная регрессия

sns.distplot(train['Fare'], kde=True)
диаграмма плотности распределения значений Series, kde отвечает за гладкую аппр-цию

sns.countplot(x='Age', data=train)
диаграмма плотности распределения значений Series, для категорийных и дискретных.

sns.barplot(y='Fare', x='AgeBand', data=titanic)
средние y в зависимости от категорийного или дискретного x.

sns.stripplot(y='Fare', x='AgeBand', data=titanic, jitter=0.4)
точки y в зависимости от катег-го или диск-го x, jitter позволяет расширить полоски.

Преобразуем

titanic['PassengerFare'] = titanic['Fare'] / titanic['TicketFreq']
пытаемся придумать новые признаки, которые позволяют различать классы.

titanic['AgeBand'] = pd.cut(titanic['Age'], [0, 13, 17, 33, 49, 60, 200], labels=False)]
пытаемся некоторые признаки 'размазать': в данном примере статистики по 34-летним почти нет, поэтому воспользуемя статистикой людей со сходным возрастом.

colOneHot = pd.get_dummies(titanic['col'], prefix='col')
titanicOneHot = pd.concat([titanic, colOneHot],
для линейных моделей важно преобразовать дискретные признаки в one-hot векторы.

train['hour'] = train['Dates'].dt.hour
если имеем цикличность по часам, то создаем признак 'hour' и размазываем по некоторым периодам.

gb = sales.groupby(['shop_id', 'date'],\ as_index=False).agg({'item_day':{'price':'mean'}})
train = pd.merge(train, gb, how='left',\ on=['shop_id', 'date']).fillna(0)
агрегирование см. future sales prediction на kaggle.
если показатели по неделям меняются плавно, то создаем признак 'наиболее частое за соседние две недели' в данном районе, в данное время суток.

train['IsItBlock'] = train['Address'].str.contains('block', case=False)
train['IsBadStreet'] = train['Address'].str.contains('IsBadStreet', case=False)
в работе с географическими данными существует неравномерность плотности распределения точек, можно работать со сгустками с помощью флажков.

train.drop_duplicates
иногда базу данных нагружают повторными

Ансамбль