МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа № 5
по курсу "Методы машинного обучения"

ИСПОЛНИТЕЛЬ: Горбовцова К.М.
ФИО
группа ИУ5-24М _____
подпись

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ:_____
ФИО

_____
подпись

"__"_____2020 г.

Москва – 2020

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.linear_model import BayesianRidge
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.metrics import r2_score
%matplotlib inline
sns.set(style="ticks")

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv("advertising.csv")
```

In [3]:
```
data.head(5)
```

Out[3]:

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
TV           200 non-null float64
Radio        200 non-null float64
Newspaper    200 non-null float64
Sales        200 non-null float64
dtypes: float64(4)
memory usage: 7.8 KB
```

In [4]:
```
data.describe()
```

Out[4]:

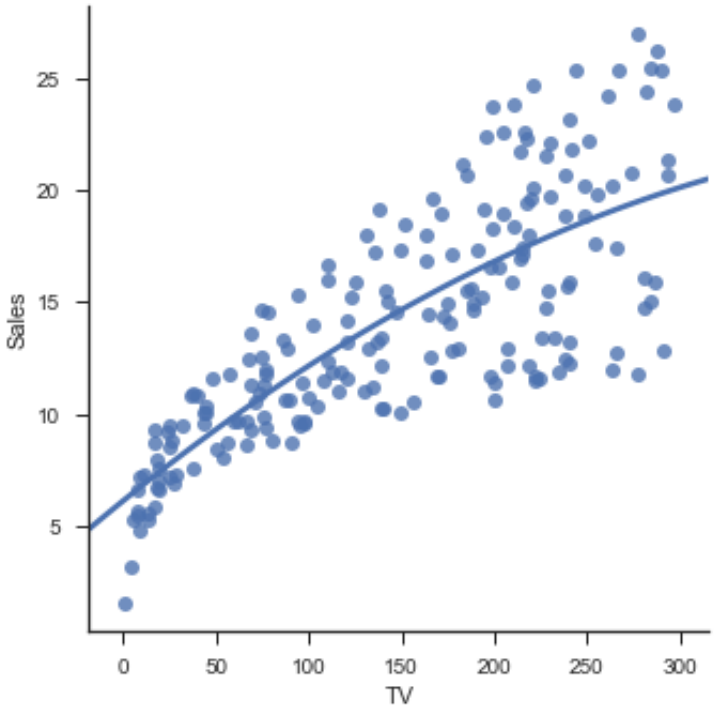
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

In [6]:
```
data.columns
```

Out[6]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

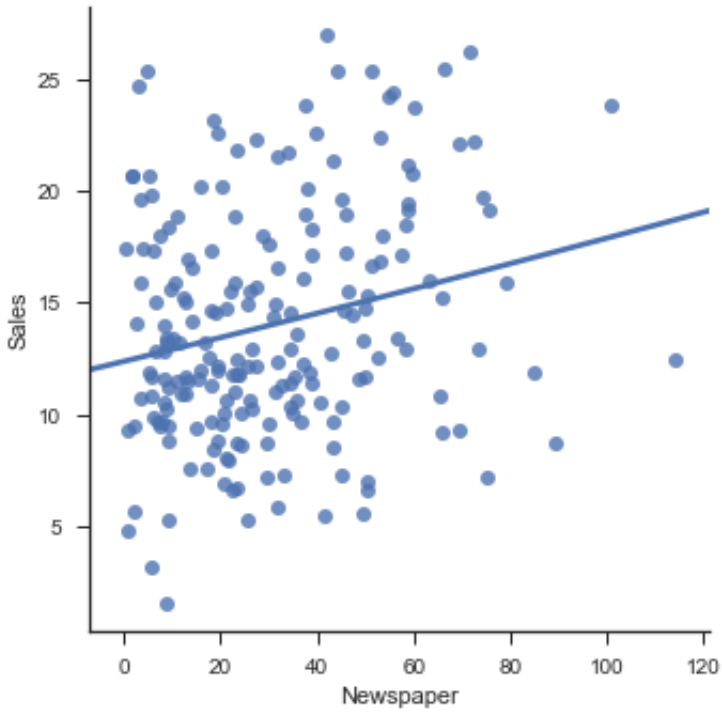
In [7]:
```
sns.lmplot(x="TV", y="Sales", data=data, order=2, ci=None)
```

Out[7]:
```
<seaborn.axisgrid.FacetGrid at 0x1129da4e0>
```



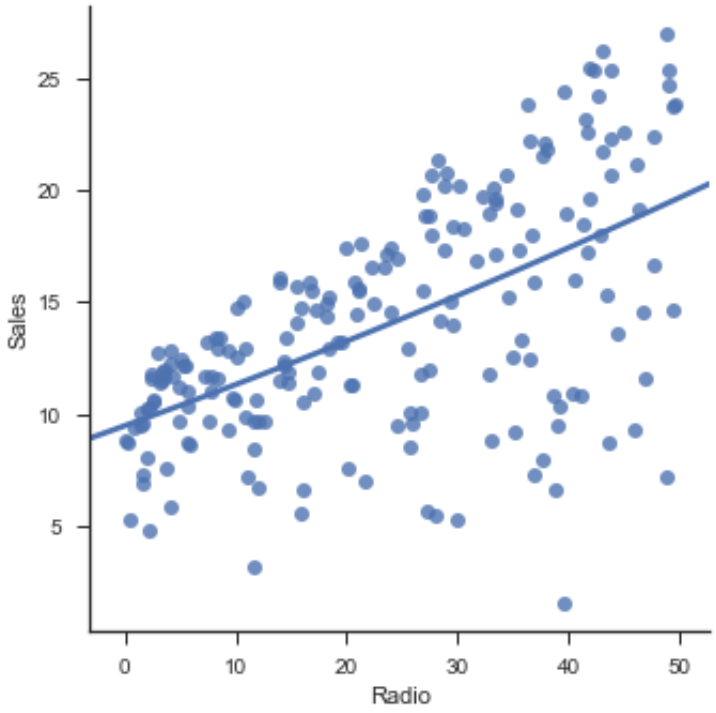
In [9]:
```
sns.lmplot(x="Newspaper", y="Sales", data=data, order=2, ci=None)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x126c80b70>



In [8]:
```
sns.lmplot(x="Radio", y="Sales", data=data, order=2, ci=None)
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0x124a2e4e0>



In [10]:
```
data.corr()
```

Out[10]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

Корреляция между TV и Sales: 0.78

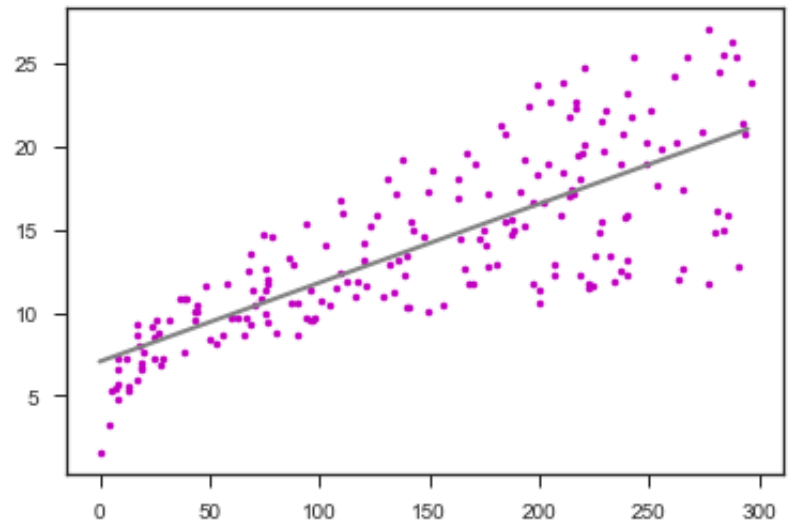
In [11]:
```
x = data["TV"].values
y = data["Sales"].values

reg = BayesianRidge(fit_intercept=True).fit(x.reshape(-1, 1), y.reshape
reg.coef_
reg.intercept_
```

Out[11]:
```
7.054854152265513
```

In [12]:
```
def func(w, b, x):
    return w*x + b
```

In [13]:
```
x_t = list(range(0, 300, 5))
y_t = [func(reg.coef_[0], reg.intercept_, x) for x in x_t]
y_tt = reg.predict(x.reshape(-1, 1))
```

In [20]:
```
plt.plot(x, y, 'm.')
plt.plot(x_t, y_t, 'grey', linewidth=2.0)
plt.show()
```



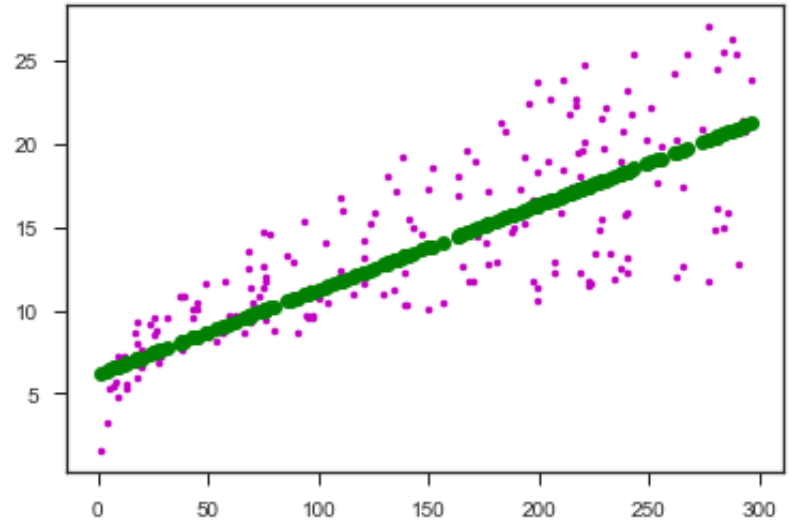
Хороший результат модели лин. регрессии

SVM

In [22]:
```
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR
```

In [36]:
```
lin_SVR = LinearSVR(C=1.0, max_iter=10000)
lin_SVR.fit(x.reshape(-1, 1), y)
predict = lin_SVR.predict(x.reshape(-1, 1))
plt.plot(x, y, 'm.')
plt.plot(x, predict, 'go')
```

Out[36]:
```
[<matplotlib.lines.Line2D at 0x128f3f278>]
```



Деревья решений

In [37]:
```
dec_tree = DecisionTreeRegressor(random_state=1, max_depth=5)
dec_tree.fit(data, data["Sales"])
dec_tree
```

Out[37]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=5,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=
None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='depreca
ted',
                      random_state=1, splitter='best')
```

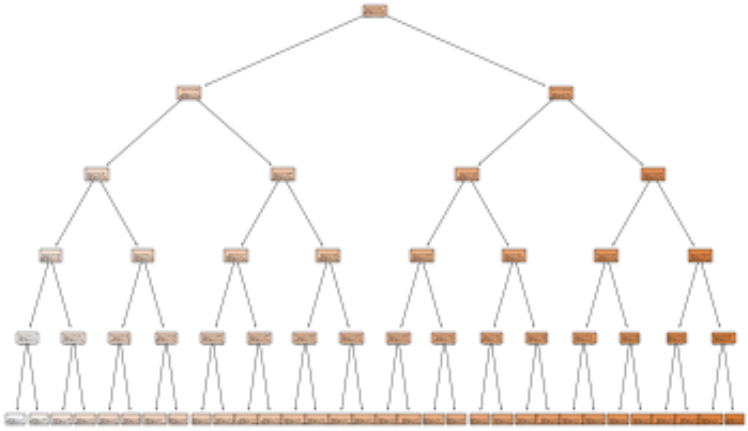
In [38]:
```
dec_predict = dec_tree.predict(data)
```

In [39]:
```
from sklearn import tree
```

```
tree.plot_tree(dec_tree, filled=True)
```

Out[39]:
```
[<matplotlib.text.Annotation at 0x12962ee80>,
 <matplotlib.text.Annotation at 0x12963cd30>,
 <matplotlib.text.Annotation at 0x12963c278>,
 <matplotlib.text.Annotation at 0x129154438>,
 <matplotlib.text.Annotation at 0x129154a20>,
 <matplotlib.text.Annotation at 0x1297df240>,
 <matplotlib.text.Annotation at 0x1297df198>,
 <matplotlib.text.Annotation at 0x129be3d68>,
 <matplotlib.text.Annotation at 0x129be3d30>,
 <matplotlib.text.Annotation at 0x129a80eb8>,
 <matplotlib.text.Annotation at 0x129a80080>,
 <matplotlib.text.Annotation at 0x129183b00>,
 <matplotlib.text.Annotation at 0x129183630>,
 <matplotlib.text.Annotation at 0x1296e8eb8>,
 <matplotlib.text.Annotation at 0x12928d470>,
 <matplotlib.text.Annotation at 0x12928db38>,
 <matplotlib.text.Annotation at 0x1296e33c8>,
 <matplotlib.text.Annotation at 0x1291544a8>,
 <matplotlib.text.Annotation at 0x129f73828>,
 <matplotlib.text.Annotation at 0x129cf0978>,
 <matplotlib.text.Annotation at 0x129cf0128>,
 <matplotlib.text.Annotation at 0x129787f98>,
 <matplotlib.text.Annotation at 0x129766f60>,
 <matplotlib.text.Annotation at 0x1296b9f98>,
 <matplotlib.text.Annotation at 0x12931ea20>,
 <matplotlib.text.Annotation at 0x1296274e0>,
 <matplotlib.text.Annotation at 0x12938eba8>,
 <matplotlib.text.Annotation at 0x129190dd8>,
 <matplotlib.text.Annotation at 0x129632b00>,
 <matplotlib.text.Annotation at 0x1298ae978>,
 <matplotlib.text.Annotation at 0x129a057f0>,
 <matplotlib.text.Annotation at 0x129deba90>,
 <matplotlib.text.Annotation at 0x12963cb38>,
 <matplotlib.text.Annotation at 0x129d08128>,
 <matplotlib.text.Annotation at 0x129bed828>,
 <matplotlib.text.Annotation at 0x129bed7b8>,
 <matplotlib.text.Annotation at 0x129bedd30>,
 <matplotlib.text.Annotation at 0x129dffb00>,
 <matplotlib.text.Annotation at 0x129c15f60>,
 <matplotlib.text.Annotation at 0x129c15048>,
 <matplotlib.text.Annotation at 0x129eb25f8>,
 <matplotlib.text.Annotation at 0x129f7dc88>,
 <matplotlib.text.Annotation at 0x129f7d400>,
 <matplotlib.text.Annotation at 0x129f850f0>,
 <matplotlib.text.Annotation at 0x129f85748>,
 <matplotlib.text.Annotation at 0x129f85588>,
 <matplotlib.text.Annotation at 0x1290f6278>,
 <matplotlib.text.Annotation at 0x1290f67f0>,
 <matplotlib.text.Annotation at 0x1290f6240>,
 <matplotlib.text.Annotation at 0x12900a2b0>,
 <matplotlib.text.Annotation at 0x12900a860>,
 <matplotlib.text.Annotation at 0x12900ae10>,
```

```
<matplotlib.text.Annotation at 0x128ffe3c8>,
<matplotlib.text.Annotation at 0x128ffe978>,

<matplotlib.text.Annotation at 0x128ffeeb8>,
<matplotlib.text.Annotation at 0x128fe3470>,
<matplotlib.text.Annotation at 0x128fe39e8>,
<matplotlib.text.Annotation at 0x128fe3f28>,
<matplotlib.text.Annotation at 0x129b7d518>,
<matplotlib.text.Annotation at 0x129b7da90>,
<matplotlib.text.Annotation at 0x129b76080>,
<matplotlib.text.Annotation at 0x129b765c0>,
```



Метрики качества

In [40]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, m

print("Метрики для линейной модели:\n")
print("Средняя абсолютная ошибка: ", mean_absolute_error(y, y_tt))
print("Средняя квадратичная ошибка: ", mean_squared_error(y, y_tt))
print("Коэффициент детерминации: ", r2_score(y, y_tt))

print("\n\nМетрики для SVM-модели:\n")
print("Средняя абсолютная ошибка: ", mean_absolute_error(y, predict))
print("Средняя квадратичная ошибка: ", mean_squared_error(y, predict))
print("Коэффициент детерминации: ", r2_score(y, predict))


print("\n\nМетрики для Decision Tree:\n")
print("Средняя абсолютная ошибка: ", mean_absolute_error(y, dec_predic
print("Средняя квадратичная ошибка: ", mean_squared_error(y, dec_predi
print("Коэффициент детерминации: ", r2_score(y, dec_predict))
```

```
Метрики для линейной модели:

Средняя абсолютная ошибка:  2.550919383216356
Средняя квадратичная ошибка:  10.512821002854928
Коэффициент детерминации:  0.6118688451058344


Метрики для SVM-модели:

Средняя абсолютная ошибка:  2.5448505255587452
Средняя квадратичная ошибка:  10.736973059051945
Коэффициент детерминации:  0.6035931980249962


Метрики для Decision Tree:

Средняя абсолютная ошибка:  0.14353164841694266
Средняя квадратичная ошибка:  0.03201810934980053
Коэффициент детерминации:  0.9988178980926156
```

Подбор гиперпараметров. Кросс-валидация

In [41]:
```
from sklearn.model_selection import cross_validate
```

In [42]:
```
scoring = {'mean': 'neg_mean_absolute_error', 'square': 'neg_mean_squa
```

In [43]:
```
scores_regr = cross_validate(BayesianRidge(fit_intercept=True),
                             x.reshape(-1, 1), y, cv=3, scoring=scoring)
scores_regr
```

Out[43]:
```
{'fit_time': array([0.00134683, 0.00257421, 0.00117993]),
 'score_time': array([0.00234008, 0.00144506, 0.00160789]),
 'test_mean': array([-2.51215213, -2.46200408, -2.76711466]),
 'test_r2': array([0.61497417, 0.65311667, 0.53715304]),
 'test_square': array([-10.83437466,  -9.33658309, -11.90833409])}
```

In [44]:
```
scores_svm = cross_validate(LinearSVR(C=1.0, max_iter=10000),
                            x.reshape(-1, 1), y, cv=3, scoring=scoring)
scores_svm
```

Out[44]:
```
{'fit_time': array([0.05261493, 0.03779411, 0.03516173]),
 'score_time': array([0.00378609, 0.00082994, 0.00082302]),
 'test_mean': array([-3.97781926, -2.43809161, -3.2147747 ]),
 'test_r2': array([0.10075056, 0.64924767, 0.33875267]),
 'test_square': array([-25.30429029,  -9.44071981, -17.01286755])}
```

In [45]:
```
scores_dec = cross_validate(DecisionTreeRegressor(random_state=1, max_
                            data, data["Sales"], cv=5, scoring=scoring)
scores_dec
```

Out[45]:
```
{'fit_time': array([0.004354  , 0.00379992, 0.00357795, 0.00352097,
0.00345826]),
 'score_time': array([0.00356698, 0.00272489, 0.002074  , 0.00266981
, 0.00261879]),
 'test_mean': array([-0.72293478, -0.7307461 , -0.66116873, -0.85487
267, -0.91550049]),
 'test_r2': array([0.97486214, 0.97589358, 0.97175881, 0.95176776, 0
.95938815]),
 'test_square': array([-0.64975012, -0.70991464, -0.63349151, -1.410
4023 , -1.08449788])}
```

In [46]:
```
print("Метрики для линейной модели:\n")
print("Средняя абсолютная ошибка: ", np.mean(scores_regr['test_mean']))
print("Средняя квадратичная ошибка: ", np.mean(scores_regr['test_square
print("Коэффициент детерминации: ", np.mean(scores_regr['test_r2']))

print("\n\nМетрики для SVM-модели:\n")
print("Средняя абсолютная ошибка: ", np.mean(scores_svm['test_mean']))
print("Средняя квадратичная ошибка: ", np.mean(scores_svm['test_square
print("Коэффициент детерминации: ", np.mean(scores_svm['test_r2']))


print("\n\nМетрики для Decision Tree:\n")
print("Средняя абсолютная ошибка: ", np.mean(scores_dec['test_mean']))
print("Средняя квадратичная ошибка: ", np.mean(scores_dec['test_square
print("Коэффициент детерминации: ", np.mean(scores_dec['test_r2']))
```

```
Метрики для линейной модели:

Средняя абсолютная ошибка:  -2.580423621885709
Средняя квадратичная ошибка:  -10.693097277894969
Коэффициент детерминации:  0.601747959666948


Метрики для SVM-модели:

Средняя абсолютная ошибка:  -3.210228524726626
Средняя квадратичная ошибка:  -17.25262588239853
Коэффициент детерминации:  0.3629169662615308


Метрики для Decision Tree:

Средняя абсолютная ошибка:  -0.7770445553321956
Средняя квадратичная ошибка:  -0.8976112886827845
Коэффициент детерминации:  0.9667340888852873
```

Оптимизация (решетчатый поиск)

In [47]:
```
from sklearn.model_selection import GridSearchCV
```

In [48]:
```
n_range = np.array(range(1,10,1))
tuned_parameters = [{'max_depth': n_range}]
tuned_parameters
```

Out[48]:
```
[{'max_depth': array([1, 2, 3, 4, 5, 6, 7, 8, 9])}]
```

In [49]:
```
%%time
clf_gs = GridSearchCV(DecisionTreeRegressor(), tuned_parameters, cv=5,
clf_gs.fit(x.reshape(-1, 1), y)
```

```
CPU times: user 98.9 ms, sys: 5.18 ms, total: 104 ms
Wall time: 110 ms
```

In [50]:
```
# Лучшая модель
clf_gs.best_estimator_
```

Out[50]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=
None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='depreca
ted',
                      random_state=None, splitter='best')
```

In [51]:
```
clf_gs.best_score_
```

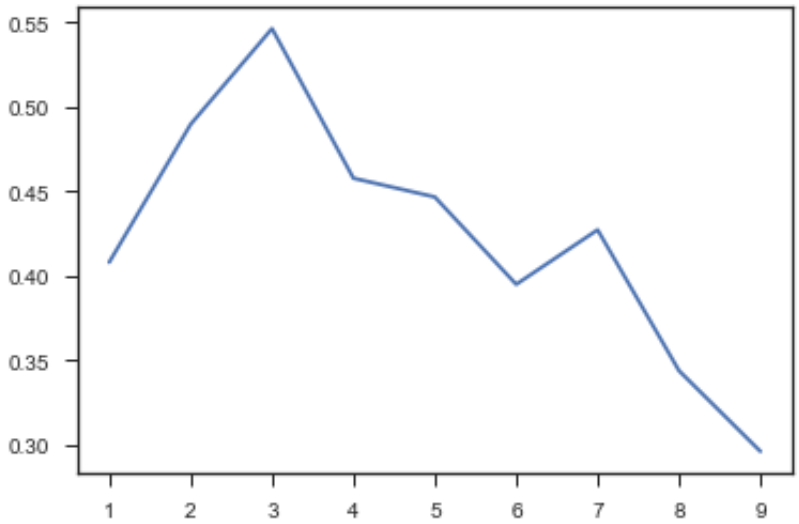
Out[51]:
```
0.5464056968965096
```

In [52]:
```
clf_gs.best_params_
```

Out[52]:
```
{'max_depth': 3}
```

In [53]:
```
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

Out[53]:
```
[<matplotlib.lines.Line2D at 0x12a0305c0>]
```



Оптимизация SVM

In [54]:
```
param_grid = {'C': [0.1,1, 10, 100], 'epsilon': [0.1, 0.2, 0.3, 0.4, 0
```

In [55]:
```
grid = GridSearchCV(LinearSVR(),param_grid,refit=True,verbose=2)
grid.fit(x.reshape(-1, 1),y)
```

```
..
[CV] ............................... C=0.1, epsilon=0.1, total=   0.
0s
[CV] C=0.1, epsilon=0.2 .............................................
..
[CV] ............................... C=0.1, epsilon=0.2, total=   0.
0s
[CV] C=0.1, epsilon=0.2 .............................................
..
[CV] ............................... C=0.1, epsilon=0.2, total=   0.
0s
[CV] C=0.1, epsilon=0.2 .............................................
..
[CV] ............................... C=0.1, epsilon=0.2, total=   0.
0s
[CV] C=0.1, epsilon=0.2 .............................................
..
[CV] ............................... C=0.1, epsilon=0.2, total=   0.
0s
[CV] C=0.1, epsilon=0.2 .............................................
```

In [56]:
```
grid.best_estimator_
```

Out[56]:
```
LinearSVR(C=10, dual=True, epsilon=0.1, fit_intercept=True,
          intercept_scaling=1.0, loss='epsilon_insensitive', max_ite
r=1000,
          random_state=None, tol=0.0001, verbose=0)
```

In [57]:
```
grid.best_score_
```

Out[57]:
```
0.5713251891027603
```

In [58]:
```
grid.best_params_
```

Out[58]:
```
{'C': 10, 'epsilon': 0.1}
```

In [59]:
```
parameters = {"alpha_1": np.logspace(-13,-5,10),
              "alpha_2": np.logspace(-9,-3,10),
              "lambda_1": np.logspace(-10,-5,10),
              "lambda_2": np.logspace(-11,-4,10)}

grid_regr = GridSearchCV(BayesianRidge(), parameters, cv=3, n_jobs=-1)
grid_regr.fit(x.reshape(-1, 1), y)
```

Out[59]:
```
GridSearchCV(cv=3, error_score=nan,
             estimator=BayesianRidge(alpha_1=1e-06, alpha_2=1e-06,
                                     alpha_init=None, compute_score=
False,
                                     copy_X=True, fit_intercept=True
,
                                     lambda_1=1e-06, lambda_2=1e-06,
                                     lambda_init=None, n_iter=300,
                                     normalize=False, tol=0.001,
                                     verbose=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'alpha_1': array([1.00000000e-13, 7.7426368
3e-13, 5.99484250e-...
                         'lambda_1': array([1.00000000e-10, 3.593813
66e-10, 1.29154967e-09, 4.64158883e-09,
       1.66810054e-08, 5.99484250e-08, 2.15443469e-07, 7.74263683e-0
7,
       2.78255940e-06, 1.00000000e-05]),
                         'lambda_2': array([1.00000000e-11, 5.994842
50e-11, 3.59381366e-10, 2.15443469e-09,
       1.29154967e-08, 7.74263683e-08, 4.64158883e-07, 2.78255940e-0
6,
       1.66810054e-05, 1.00000000e-04])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score
=False,
             scoring=None, verbose=0)
```

In [60]:
```
grid_regr.best_estimator_
```

Out[60]:
```
BayesianRidge(alpha_1=1e-05, alpha_2=1e-09, alpha_init=None,
              compute_score=False, copy_X=True, fit_intercept=True,
              lambda_1=1e-10, lambda_2=0.0001, lambda_init=None, n_i
ter=300,
              normalize=False, tol=0.001, verbose=False)
```

In [61]:
```
grid_regr.best_score_
```

Out[61]:
```
0.6017531508217578
```

In [62]:
```
grid_regr.best_params_
```

Out[62]:
```
{'alpha_1': 1e-05, 'alpha_2': 1e-09, 'lambda_1': 1e-10, 'lambda_2':
0.0001}
```

In [63]:
```
reg = BayesianRidge(fit_intercept=True, alpha_1=1e-05, alpha_2=1e-09, 
y_tt = reg.predict(x.reshape(-1, 1))

lin_SVR = LinearSVR(C=1.0, max_iter=10000, epsilon=1.0)
lin_SVR.fit(x.reshape(-1, 1), y)
predict = lin_SVR.predict(x.reshape(-1, 1))

dec_tree = DecisionTreeRegressor(random_state=1, max_depth=3)
dec_tree.fit(data, data["Sales"])
dec_predict = dec_tree.predict(data)
```

In [64]:
```
print("Метрики для линейной модели:\n")
print("Средняя абсолютная ошибка: ", mean_absolute_error(y, y_tt))
print("Средняя квадратичная ошибка: ", mean_squared_error(y, y_tt))
print("Коэффициент детерминации: ", r2_score(y, y_tt))

print("\n\nМетрики для SVM-модели:\n")
print("Средняя абсолютная ошибка: ", mean_absolute_error(y, predict))
print("Средняя квадратичная ошибка: ", mean_squared_error(y, predict))
print("Коэффициент детерминации: ", r2_score(y, predict))


print("\n\nМетрики для Decision Tree:\n")
print("Средняя абсолютная ошибка: ", mean_absolute_error(y, dec_predic
print("Средняя квадратичная ошибка: ", mean_squared_error(y, dec_predi
print("Коэффициент детерминации: ", r2_score(y, dec_predict))
```

```
Метрики для линейной модели:

Средняя абсолютная ошибка:  2.5508292802546
Средняя квадратичная ошибка:  10.512794897173503
Коэффициент детерминации:  0.6118698089221382


Метрики для SVM-модели:

Средняя абсолютная ошибка:  2.6278521710815728
Средняя квадратичная ошибка:  11.514845266318599
Коэффициент детерминации:  0.5748743186600296


Метрики для Decision Tree:

Средняя абсолютная ошибка:  0.7095532407407409
Средняя квадратичная ошибка:  0.7222188657407407
Коэффициент детерминации:  0.9733358303760538
```

Подбор параметров улучшил показатели моделей