In [43]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
```

In [44]:
```
housing_data = pd.read_excel('AmesHousing.xls')
housing_data.head()
```

Out[44]:

| | Order | PID | MS SubClass | MS Zoning | Lot Frontage | Lot Area | Street | Alley | Lot Shape | Land Contour | ... | Pc Ar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 526301100 | 20 | RL | 141.0 | 31770 | Pave | NaN | IR1 | Lvl | ... | |
| **1** | 2 | 526350040 | 20 | RH | 80.0 | 11622 | Pave | NaN | Reg | Lvl | ... | |
| **2** | 3 | 526351010 | 20 | RL | 81.0 | 14267 | Pave | NaN | IR1 | Lvl | ... | |
| **3** | 4 | 526353030 | 20 | RL | 93.0 | 11160 | Pave | NaN | Reg | Lvl | ... | |
| **4** | 5 | 527105010 | 60 | RL | 74.0 | 13830 | Pave | NaN | IR1 | Lvl | ... | |

5 rows × 82 columns

In [45]:
```
housing_data.dtypes
```

Out[45]:

| | |
|---|---|
| Order | int64 |
| PID | int64 |
| MS SubClass | int64 |
| MS Zoning | object |
| Lot Frontage | float64 |
| Lot Area | int64 |
| Street | object |
| Alley | object |
| Lot Shape | object |
| Land Contour | object |
| Utilities | object |
| Lot Config | object |
| Land Slope | object |
| Neighborhood | object |
| Condition 1 | object |
| Condition 2 | object |
| Bldg Type | object |
| House Style | object |
| Overall Qual | int64 |
| Overall Cond | int64 |
| Year Built | int64 |
| Year Remod/Add | int64 |
| Roof Style | object |
| Roof Matl | object |
| Exterior 1st | object |
| Exterior 2nd | object |
| Mas Vnr Type | object |
| Mas Vnr Area | float64 |
| Exter Qual | object |
| Exter Cond | object |
| ... | |
| Bedroom AbvGr | int64 |
| Kitchen AbvGr | int64 |
| Kitchen Qual | object |
| TotRms AbvGrd | int64 |
| Functional | object |
| Fireplaces | int64 |
| Fireplace Qu | object |
| Garage Type | object |
| Garage Yr Blt | float64 |
| Garage Finish | object |
| Garage Cars | float64 |
| Garage Area | float64 |
| Garage Qual | object |
| Garage Cond | object |
| Paved Drive | object |
| Wood Deck SF | int64 |
| Open Porch SF | int64 |
| Enclosed Porch | int64 |
| 3Ssn Porch | int64 |
| Screen Porch | int64 |
| Pool Area | int64 |
| Pool QC | object |
| Fence | object |
| Misc Feature | object |
| Misc Val | int64 |
| Mo Sold | int64 |

```
Yr Sold             int64
Sale Type          object
Sale Condition     object
SalePrice           int64
Length: 82, dtype: object
```

In [46]:
```
housing_data.drop(columns=['Order', 'PID'], inplace=True)
```

In [47]:
```
housing_data.dtypes
```

Out[47]:
```
MS SubClass         int64
MS Zoning          object
Lot Frontage      float64
Lot Area            int64
Street             object
Alley              object
Lot Shape          object
Land Contour       object
Utilities          object
Lot Config         object
Land Slope         object
Neighborhood       object
Condition 1        object
Condition 2        object
Bldg Type          object
House Style        object
Overall Qual        int64
Overall Cond        int64
Year Built          int64
Year Remod/Add      int64
Roof Style         object
Roof Matl          object
Exterior 1st       object
Exterior 2nd       object
Mas Vnr Type       object
Mas Vnr Area      float64
Exter Qual         object
Exter Cond         object
Foundation         object
Bsmt Qual          object
                   ...
Bedroom AbvGr       int64
Kitchen AbvGr       int64
Kitchen Qual       object
TotRms AbvGrd       int64
Functional         object
Fireplaces          int64
Fireplace Qu       object
Garage Type        object
Garage Yr Blt     float64
Garage Finish      object
Garage Cars       float64
Garage Area       float64
Garage Qual        object
Garage Cond        object
Paved Drive        object
Wood Deck SF        int64
Open Porch SF       int64
Enclosed Porch      int64
3Ssn Porch          int64
Screen Porch        int64
Pool Area           int64
Pool QC            object
Fence              object
Misc Feature       object
Misc Val            int64
Mo Sold             int64
```

```
Yr Sold             int64
Sale Type          object
Sale Condition     object
SalePrice           int64
Length: 80, dtype: object
```

In [48]:
```
import warnings
warnings.filterwarnings("ignore")
housing_continuous = housing_data.select_dtypes (include = ['int64','float64'
])
#housing_continuous.drop(columns = ['MS SubClass','Year Built','Year Remod/Ad
d','Mo Sold','Yr Sold'],inplace = True)
housing_continuous.drop(columns = ['MS SubClass','Mo Sold'],inplace = True)
#choosing continuous columns based on d-type but dropping some of them based o
n looking at the description
#Continuing to keep year related columns as continuous to reduce dimensionalit
y
```

In [49]:
```
housing_continuous.shape
```

Out[49]:
```
(2930, 35)
```

In [50]:
```
cont_name = list(housing_continuous)
```

In [51]:
```
cat_name = list(set(housing_data.columns)- set(housing_continuous.columns))
```

In [52]:
```
len(cat_name)
```

Out[52]:
```
45
```

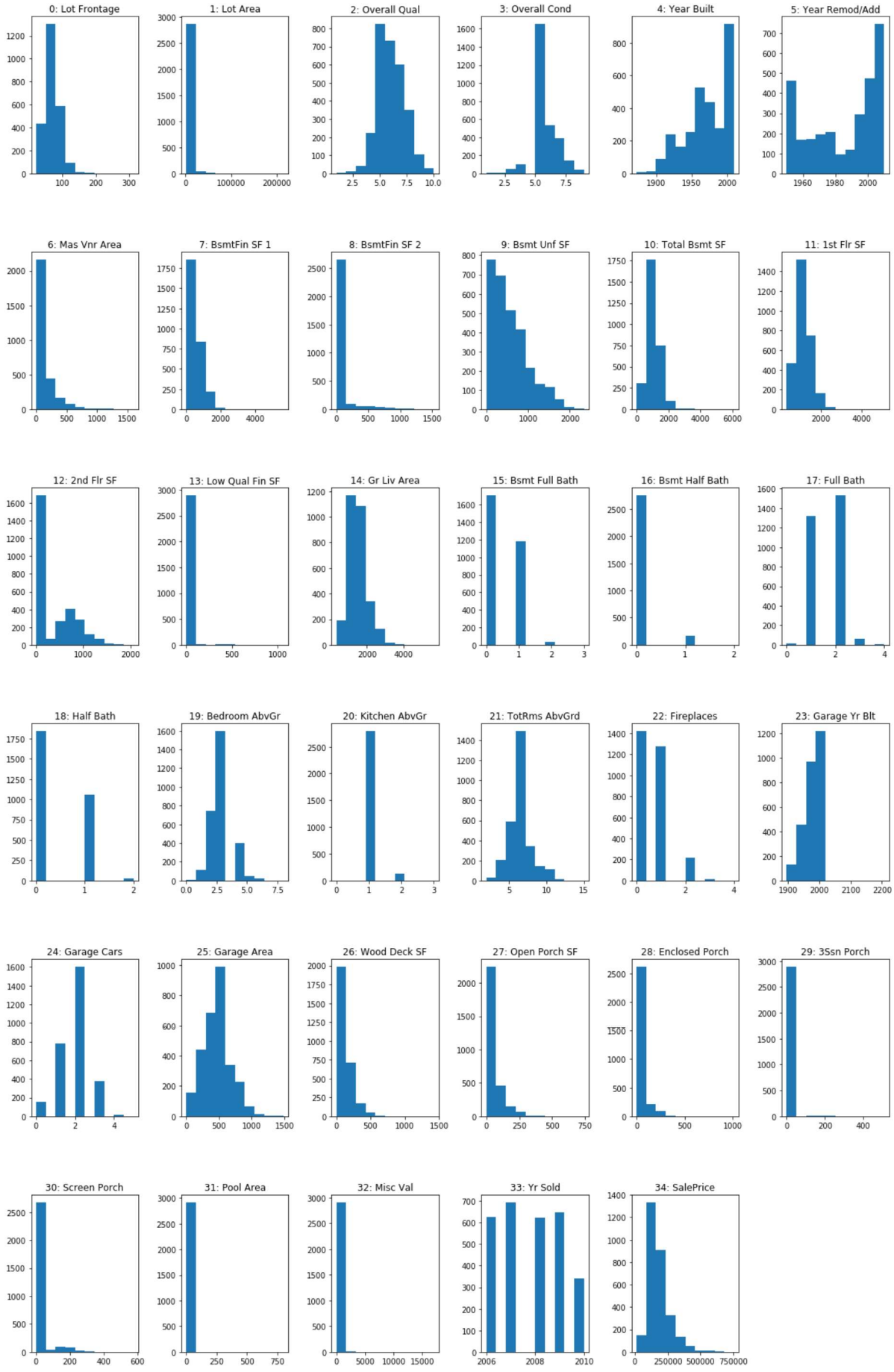
In [53]:
```
cat_name
```

Out[53]:
```
['Central Air',
 'Bsmt Qual',
 'Paved Drive',
 'Garage Qual',
 'Bsmt Cond',
 'Garage Type',
 'Land Contour',
 'Street',
 'House Style',
 'Heating QC',
 'Fence',
 'Heating',
 'Mo Sold',
 'Roof Matl',
 'Condition 1',
 'Exter Cond',
 'Neighborhood',
 'Alley',
 'Mas Vnr Type',
 'MS Zoning',
 'Lot Config',
 'Misc Feature',
 'Exterior 1st',
 'BsmtFin Type 1',
 'Exterior 2nd',
 'Garage Cond',
 'Utilities',
 'Functional',
 'Roof Style',
 'Exter Qual',
 'Sale Type',
 'Fireplace Qu',
 'Sale Condition',
 'Garage Finish',
 'Condition 2',
 'Electrical',
 'Pool QC',
 'Bsmt Exposure',
 'BsmtFin Type 2',
 'Bldg Type',
 'Foundation',
 'Lot Shape',
 'MS SubClass',
 'Kitchen Qual',
 'Land Slope']
```

In [54]:
```
#housing_categorical = housing_data.select_dtypes (include = ['object'])
#housing_categorical['MS SubClass'] = housing_data['MS SubClass']
```

# 1.1

In [55]:
```
fig, axes = plt.subplots (6,6,figsize = (20,32))
plt.subplots_adjust(wspace=0.40,hspace=0.50)

for i, ax in enumerate (axes.ravel()):
    if i > 34:
        ax.set_visible(False)
        continue
    ax.hist(housing_continuous.iloc[:,i])
    ax.set_title("{}: {}".format(i, cont_name[i]))
```

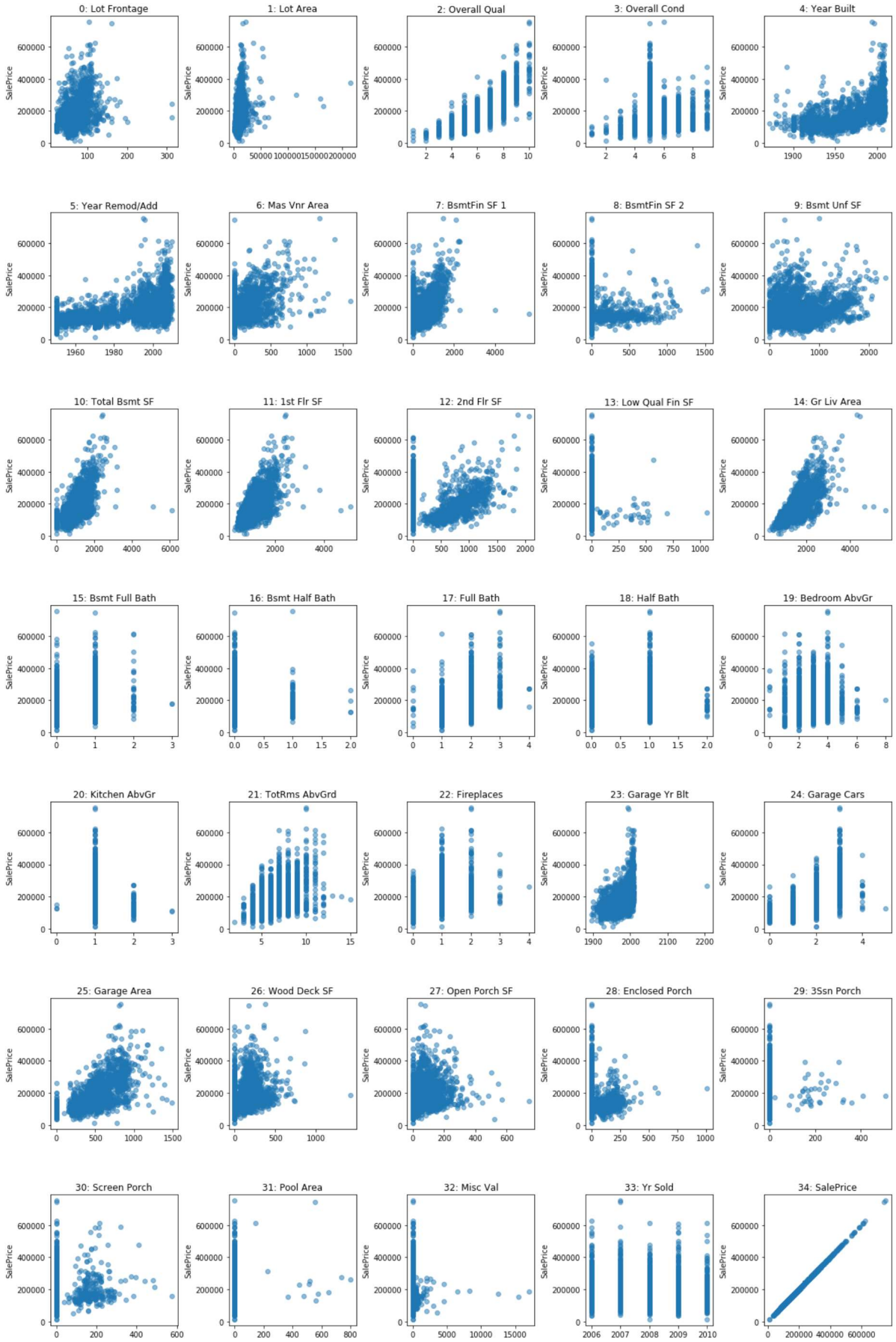

0: Lot Frontage
1: Lot Area
2: Overall Qual
3: Overall Cond
4: Year Built
5: Year Remod/Add
6: Mas Vnr Area
7: BsmtFin SF 1
8: BsmtFin SF 2
9: Bsmt Unf SF
10: Total Bsmt SF
11: 1st Flr SF
12: 2nd Flr SF
13: Low Qual Fin SF
14: Gr Liv Area
15: Bsmt Full Bath
16: Bsmt Half Bath
17: Full Bath
18: Half Bath
19: Bedroom AbvGr
20: Kitchen AbvGr
21: TotRms AbvGrd
22: Fireplaces
23: Garage Yr Blt
24: Garage Cars
25: Garage Area
26: Wood Deck SF
27: Open Porch SF
28: Enclosed Porch
29: 3Ssn Porch
30: Screen Porch
31: Pool Area
32: Misc Val
33: Yr Sold
34: SalePrice

- We can see that all variables are not continuous - some of the histogram distribution have a lot of spaces in between them indicating that they are discrete (ex: Full bath, Garage Cars etc.)

*A handful variables seem to have outliers and that is why we see that the prominent histograms are either to the left/ right along with a few smaller ones distributed across (ex:BsmtFin SF 2 etc.)

# 1.2

In [56]:
```
fig, axes = plt.subplots(7, 5, figsize=(20, 32))
plt.subplots_adjust(wspace=0.40,hspace=0.50)

for i, ax in enumerate(axes.ravel()):
    if i > 34:
        ax.set_visible(False)
        continue
    ax.plot(housing_continuous.iloc[:, i], housing_continuous['SalePrice'],
'o', alpha=.5)
    ax.set_title("{}: {}".format(i, cont_name[i]))
    ax.set_ylabel("SalePrice")
```




In [57]:
```
housing_data[cat_name] = housing_data[cat_name].replace({np.nan:'NA'})
#Taking nan values as NA, a separate category
```

# 1.3

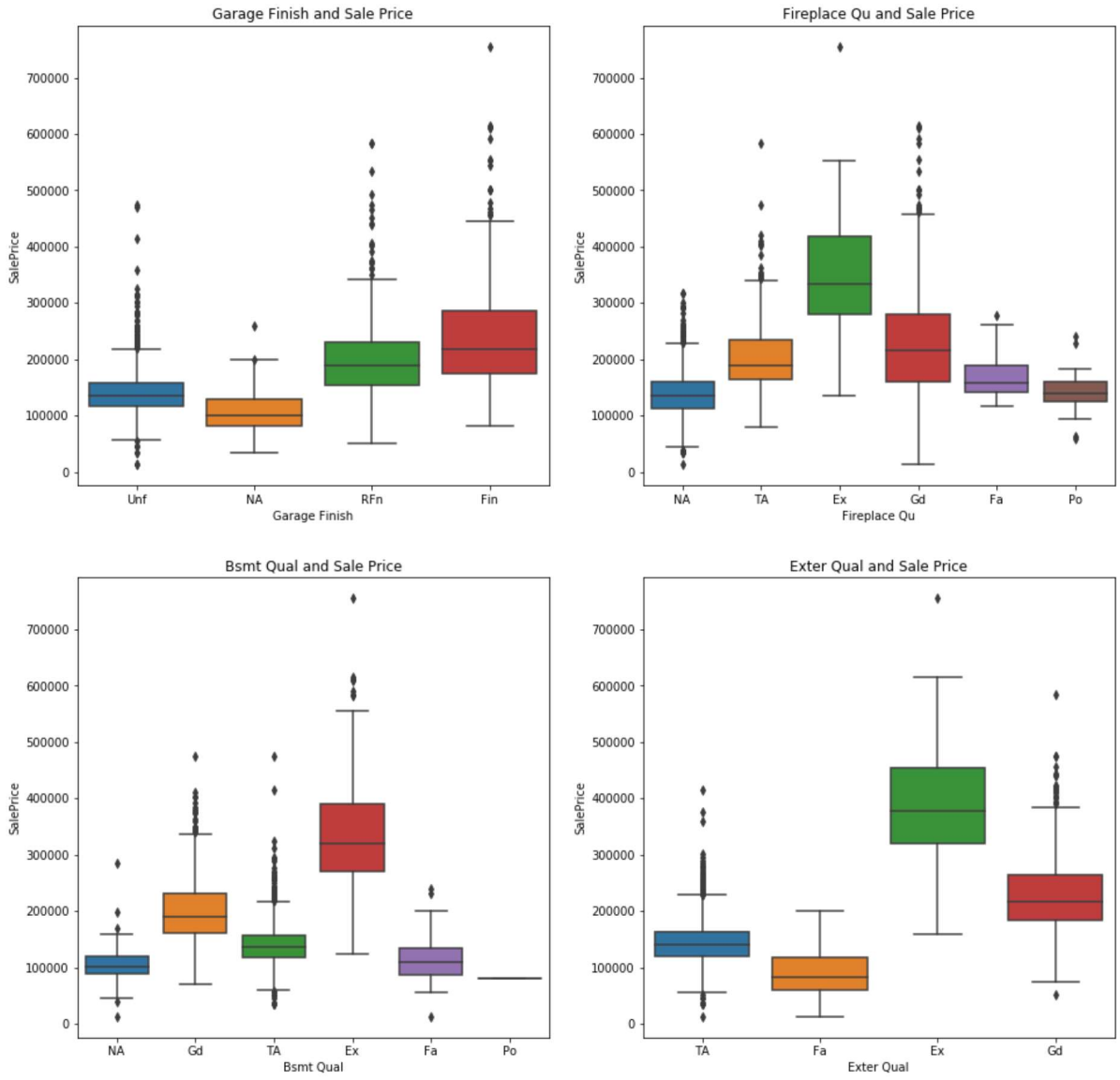
In [58]:
```
from sklearn.model_selection import train_test_split
cont_name.remove('SalePrice')
X_train, X_test, y_train, y_test = train_test_split (housing_data.loc[:, housi
ng_data.columns != 'SalePrice'],housing_data['SalePrice'], random_state = 0)
X_train = pd.concat((X_train[cat_name], X_train[cont_name]), axis=1)
```

In [59]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
cat_score = []

for i in cat_name:
    scores = cross_val_score(LinearRegression(), pd.get_dummies(X_train[i]), y
_train, cv=5)
    cat_score.append(np.mean(scores))

print ('The top 5 R^2 values are')

for i in range(0,5):

    print(cat_name[np.argsort(cat_score)[-5:][i]], cat_score[np.argsort(cat_sc
ore)[-5:][i]])
```

```
The top 5 R^2 values are
Garage Type 0.23464024676523776
Foundation 0.27072441117959734
Garage Finish 0.30005099379138944
Fireplace Qu 0.31984124628027705
Exter Qual 0.5219081154566937
```

In [60]:
```
#Plotting the top 4 values from the previous section
import seaborn as sns
train_set = pd.concat([X_train, y_train], axis=1)
fig, ax = plt.subplots(2,2, figsize=(16,16))

_ = sns.boxplot(x='Garage Finish', y='SalePrice',data= train_set, ax=ax[0][0])
ax[0,0].set_title('Garage Finish and Sale Price')

_ = sns.boxplot(x='Fireplace Qu', y='SalePrice',data= train_set, ax=ax[0][1])
ax[0,1].set_title('Fireplace Qu and Sale Price')

_ = sns.boxplot(x='Bsmt Qual', y='SalePrice',data= train_set, ax=ax[1][0])
ax[1,0].set_title('Bsmt Qual and Sale Price')

_ = sns.boxplot(x='Exter Qual', y='SalePrice',data= train_set, ax=ax[1][1])
ax[1,1].set_title('Exter Qual and Sale Price')
```

Out[60]: Text(0.5, 1.0, 'Exter Qual and Sale Price')



In [61]:
```
X_train
```

Out[61]:

| | Central Air | Bsmt Qual | Paved Drive | Garage Qual | Bsmt Cond | Garage Type | Land Contour | Street | House Style | Heating QC | ... | Gar C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **896** | N | NA | N | TA | NA | Detchd | Lvl | Pave | 1.5Fin | TA | ... | |
| **1901** | N | NA | N | NA | NA | NA | Low | Pave | 1Story | Fa | ... | |
| **390** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | TA | ... | |
| **1293** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 1.5Fin | Gd | ... | |
| **1767** | Y | Ex | Y | TA | TA | Attchd | Lvl | Pave | 2Story | Ex | ... | |
| **251** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **2847** | Y | Fa | Y | Fa | TA | Detchd | Lvl | Pave | 1.5Fin | TA | ... | |
| **357** | Y | Gd | Y | TA | Gd | BuiltIn | Lvl | Pave | 2Story | Gd | ... | |
| **1396** | Y | TA | N | TA | TA | 2Types | Lvl | Pave | 1.5Fin | Gd | ... | |
| **852** | Y | Gd | Y | NA | TA | NA | HLS | Pave | 1Story | TA | ... | |
| **841** | Y | Gd | Y | TA | Gd | Detchd | Lvl | Pave | 1Story | Ex | ... | |
| **1756** | Y | Gd | Y | TA | TA | BuiltIn | Lvl | Pave | SLvl | Ex | ... | |
| **2472** | Y | Gd | Y | NA | TA | NA | Lvl | Pave | 1Story | Ex | ... | |
| **528** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **232** | Y | Gd | Y | TA | TA | Attchd | Low | Pave | 1Story | TA | ... | |
| **610** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | SLvl | TA | ... | |
| **1380** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **2074** | Y | TA | Y | TA | Gd | Detchd | Lvl | Pave | 1Story | Ex | ... | |
| **1043** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 2Story | TA | ... | |
| **1374** | Y | NA | N | TA | NA | Detchd | Lvl | Pave | 1.5Fin | TA | ... | |
| **653** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 1Story | Ex | ... | |
| **1387** | Y | TA | Y | NA | TA | NA | Lvl | Pave | 1Story | TA | ... | |
| **1703** | Y | Ex | Y | TA | TA | BuiltIn | Lvl | Pave | 2Story | Ex | ... | |
| **676** | N | NA | Y | TA | NA | Detchd | Lvl | Pave | 1Story | Fa | ... | |
| **1174** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 2Story | Ex | ... | |
| **969** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Gd | ... | |
| **1042** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 2Story | TA | ... | |
| **766** | N | TA | Y | TA | TA | Detchd | Lvl | Pave | 2Story | TA | ... | |
| **1118** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 2Story | Ex | ... | |
| **2106** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2496** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 1Story | TA | ... | |
| **1871** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Gd | ... | |
| **2046** | N | TA | P | TA | TA | Detchd | Lvl | Pave | 1.5Fin | Fa | ... | |

| | Central Air | Bsmt Qual | Paved Drive | Garage Qual | Bsmt Cond | Garage Type | Land Contour | Street | House Style | Heating QC | ... | Gar C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **755** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 1.5Fin | TA | ... | |
| **976** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **2163** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **1940** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Gd | ... | |
| **2825** | Y | Gd | Y | TA | TA | Basment | Lvl | Pave | SLvl | Gd | ... | |
| **2120** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Gd | ... | |
| **2893** | Y | Gd | Y | TA | TA | Attchd | Low | Pave | 1Story | TA | ... | |
| **537** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **1701** | Y | Ex | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **2897** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 2Story | Ex | ... | |
| **2222** | Y | Gd | Y | TA | TA | Basment | Bnk | Pave | 1Story | TA | ... | |
| **2135** | Y | Gd | Y | TA | TA | Detchd | Lvl | Pave | SLvl | TA | ... | |
| **2599** | Y | Fa | Y | Fa | Fa | Attchd | Lvl | Pave | 1.5Fin | Ex | ... | |
| **705** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 1.5Fin | TA | ... | |
| **2362** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 1Story | TA | ... | |
| **2648** | Y | TA | Y | TA | TA | Detchd | Lvl | Pave | 1Story | Ex | ... | |
| **2647** | Y | Fa | N | Fa | Fa | Detchd | Lvl | Pave | 2Story | TA | ... | |
| **1828** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 2Story | TA | ... | |
| **1778** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |
| **277** | N | TA | P | TA | TA | Detchd | Lvl | Pave | 1.5Fin | Ex | ... | |
| **1033** | Y | Gd | Y | TA | TA | Detchd | Lvl | Pave | 2Story | TA | ... | |
| **1731** | Y | Gd | Y | TA | TA | Attchd | Lvl | Pave | 2Story | Ex | ... | |
| **763** | Y | Gd | Y | NA | Gd | NA | Lvl | Pave | 1Story | TA | ... | |
| **835** | Y | Gd | Y | TA | TA | Detchd | Low | Pave | 1Story | Gd | ... | |
| **1653** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | SLvl | TA | ... | |
| **2607** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | SLvl | TA | ... | |
| **2732** | Y | TA | Y | TA | TA | Attchd | Lvl | Pave | 1Story | Ex | ... | |

2197 rows × 79 columns

# 1.4

#1.4 - a

In [62]:
```
from sklearn.compose import make_column_transformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder

preprocess_imputer = ColumnTransformer([('ContImpute', SimpleImputer(strategy=
'median'), cont_name),
    ('Ohe', OneHotEncoder(handle_unknown='ignore'), cat_name)], remainder='pas
sthrough')
```

In [63]:
```
import warnings
warnings.filterwarnings("ignore")

linreg_model = make_pipeline(preprocess_imputer, LinearRegression())
linreg_scores = cross_val_score(linreg_model, X_train, y_train, cv=5)

lasso_model = make_pipeline (preprocess_imputer, Lasso())
lasso_scores = cross_val_score (lasso_model, X_train, y_train, cv=5)

ridge_model = make_pipeline (preprocess_imputer, Ridge())
ridge_scores = cross_val_score (ridge_model, X_train, y_train, cv=5)

elastic_model = make_pipeline(preprocess_imputer, ElasticNet())
elastic_scores = cross_val_score(elastic_model, X_train, y_train, cv=5)


print('Linear Regression Initial Score ', np.mean(linreg_scores))
print ('Lasso Regression Initial Score ', np.mean(lasso_scores))
print ('Ridge Regression Initial Score ', np.mean(ridge_scores))
print('Elastic Net Initial Score ', np.mean(elastic_scores))
```

```
Linear Regression Initial Score  0.8410318378923524
Lasso Regression Initial Score  0.8735966862603408
Ridge Regression Initial Score  0.6964829199408656
Elastic Net Initial Score  0.8437582497513526
```

#1.4 - b (Scaling included)

In [64]:
```
from sklearn.preprocessing import StandardScaler

scaling = ColumnTransformer([('scaling', StandardScaler(), cont_name)], remain
der='passthrough')

preprocess_scaling = ColumnTransformer([('ContImpute', SimpleImputer(strategy=
'median'), np.arange(0,len(cont_name))),
    ('Ohe', OneHotEncoder(handle_unknown='ignore'), np.arange(len(cont_name),l
en(X_train.columns)))], remainder='passthrough')
```

In [65]:
```
linreg_model_scaling = make_pipeline(scaling, preprocess_scaling, LinearRegres
sion())
linreg_scores_scaling = cross_val_score(linreg_model_scaling, X_train, y_train
, cv=5)

lasso_model_scaling = make_pipeline (scaling, preprocess_scaling, Lasso())
lasso_scores_scaling = cross_val_score (lasso_model_scaling, X_train, y_train,
cv=5)

ridge_model_scaling = make_pipeline (scaling, preprocess_scaling, Ridge())
ridge_scores_scaling = cross_val_score (ridge_model_scaling, X_train, y_train,
cv=5)

elastic_model_scaling = make_pipeline(preprocess_imputer, ElasticNet())
elastic_scores_scaling = cross_val_score(elastic_model, X_train, y_train, cv=5
)

print('Linear Regression (Scaling) ', np.mean(linreg_scores_scaling))
print ('Lasso Regression (Scaling) ', np.mean(lasso_scores_scaling))
print ('Ridge Regression (Scaling) ', np.mean(ridge_scores_scaling))
print('Elastic Net (Scaling) ', np.mean(elastic_scores_scaling))
```

```
Linear Regression (Scaling)  0.8348515787662137
Lasso Regression (Scaling)  0.8736450936390707
Ridge Regression (Scaling)  0.8707794944172409
Elastic Net (Scaling)  0.8437582497513526
```

# Scaling definitely helps in the case of ridge regression as we see the score move from 0.68 to 0.86 on scaling. The impact is visible but not huge for the other models but it might be due to the fact that they show a ~85% score which is high initially itself.

# 1.5

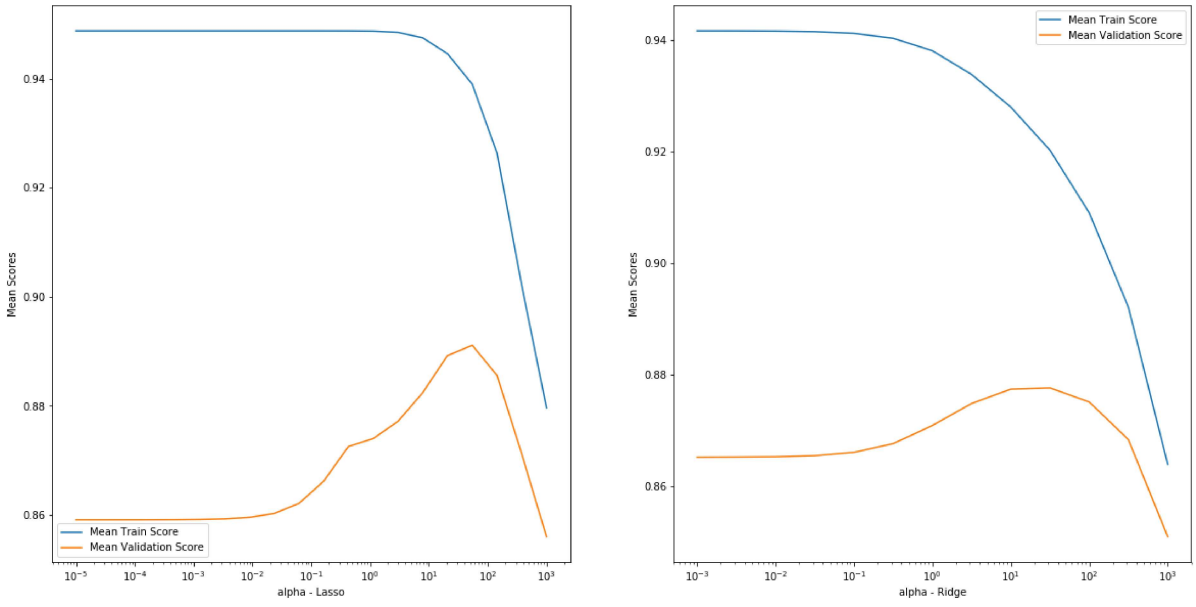
In [66]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
```

In [68]:
```
#paragrid_linreg = {'linreg__fit_intercept':[True,False], 'linreg__normalize':
[True,False], 'linreg__copy_X':[True, False]}
paragrid_lasso = {'lasso__alpha': np.logspace(-5, 3, 20)}
paragrid_ridge = {'ridge__alpha': np.logspace(-3, 3, 13)}
paragrid_elastic = {'elastic__alpha': np.logspace(-3, 3, 13), 'elastic__l1_rat
io':[0.01, .1, .4, .7, .9, 0.95, 1]}


#pipeline_linreg = Pipeline(steps=[('scaling',scaling), ('preprocess', preproc
ess_scaling), ('linreg', LinearRegression())])
pipeline_lasso = Pipeline(steps=[('scaling',scaling), ('preprocess', preproces
s_scaling), ('lasso', Lasso())])
pipeline_ridge = Pipeline(steps=[('scaling',scaling), ('preprocess', preproces
s_scaling), ('ridge', Ridge())])
pipeline_elastic = Pipeline(steps=[('scaling',scaling), ('preprocess', preproc
ess_scaling), ('elastic', ElasticNet())])


#gcv_linreg = GridSearchCV(pipeline_linreg, param_grid=paragrid_linreg, cv=5,
 return_train_score=True)
#gcv_linreg.fit(X_train, y_train)

gcv_lasso = GridSearchCV(pipeline_lasso, param_grid= paragrid_lasso, cv=5, ret
urn_train_score=True)
gcv_lasso.fit(X_train, y_train)

gcv_ridge = GridSearchCV(pipeline_ridge, param_grid=paragrid_ridge, cv=5, retu
rn_train_score=True)
gcv_ridge.fit(X_train, y_train)

gcv_elastic = GridSearchCV(pipeline_elastic, param_grid=paragrid_elastic, cv=5
, return_train_score=True)
gcv_elastic.fit(X_train, y_train)

#print("Linear regression best parameter is".format(gcv_linreg.best_params_))
print("Lasso best parameter is {}".format(gcv_lasso.best_params_))
print("Ridge best parameters is {}".format(gcv_ridge.best_params_))
print("Elastic Net best parameters is {}".format(gcv_elastic.best_params_))

#These values change on running through this multiple times, probably due to s
eed issues.
#I had these at 141, 53, <50, and 0.9) one day before submission
```

```
Lasso best parameter is {'lasso__alpha': 54.555947811685144}
Ridge best parameters is {'ridge__alpha': 31.622776601683793}
Elastic Net best parameters is {'elastic__alpha': 31.622776601683793, 'elasti
c__l1_ratio': 1}
```

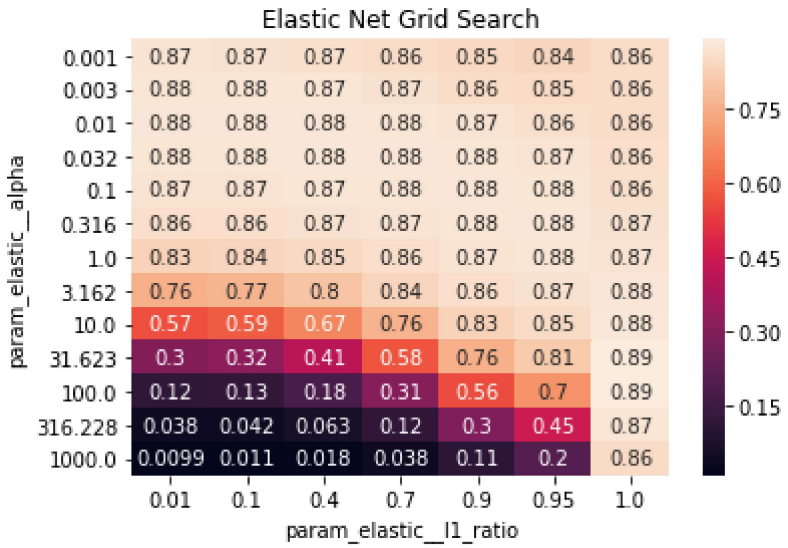
In [69]:
```
#Visualising the performance of Lasso and Ridge

fig, ax = plt.subplots(1,2, figsize=(20,10))

lasso_res = pd.DataFrame(gcv_lasso.cv_results_)
ridge_res = pd.DataFrame (gcv_ridge.cv_results_)

ax[0].plot(lasso_res['param_lasso__alpha'], lasso_res['mean_train_score'], lab
el = 'Mean Train Score')
ax[0].plot(lasso_res['param_lasso__alpha'], lasso_res['mean_test_score'], labe
l = 'Mean Validation Score')

ax[1].plot(ridge_res['param_ridge__alpha'], ridge_res['mean_train_score'], lab
el = 'Mean Train Score')
ax[1].plot(ridge_res['param_ridge__alpha'], ridge_res['mean_test_score'], labe
l = 'Mean Validation Score')

ax[0].set_xscale('log')
ax[1].set_xscale('log')

ax[0].legend()
ax[1].legend()

ax[0].set_xlabel('alpha - Lasso')
ax[0].set_ylabel('Mean Scores')

ax[1].set_xlabel('alpha - Ridge')
ax[1].set_ylabel('Mean Scores')
```

Out[69]: Text(0, 0.5, 'Mean Scores')



In [70]:
```
#Visualising the performance of ElasticNet
import seaborn as sns

res = pd.pivot_table(pd.DataFrame(gcv_elastic.cv_results_), values = 'mean_tes
t_score',index = 'param_elastic__alpha',
                     columns = 'param_elastic__l1_ratio')

res = res.set_index(res.index.values.round(3))
ax = sns.heatmap(res, annot = True)
plt.title('Elastic Net Grid Search')
plt.ylabel ('param_elastic__alpha')
```

Out[70]: Text(33.0, 0.5, 'param_elastic__alpha')




# 1.6

In [71]:
```
#Checking number of co-efficient after Lasso
len (gcv_lasso.best_estimator_.named_steps['lasso'].coef_)
#Checking number of co-efficients after onehotencoding
#len(cont_name) = 34
x = scaling.fit_transform(X_train)
categorical_ohe_lasso = preprocess_scaling.fit(x).named_transformers_['Ohe'].g
et_feature_names()
len(preprocess_scaling.fit(x).named_transformers_['Ohe'].get_feature_names())
#verifying that lengths are the same
```

Out[71]: 307

In [72]:
```
len(np.append (cont_name, preprocess_scaling.fit(x).named_transformers_['Ohe']
.get_feature_names()))
#verifying that lengths are the same
```

Out[72]: 341

In [73]:
```
lasso_features = pd.DataFrame(gcv_lasso.best_estimator_.named_steps['lasso'].c
oef_,np.append (cont_name, preprocess_scaling.fit(x).named_transformers_['Ohe'
].get_feature_names())).reset_index()
lasso_features.columns = ['lasso_feature','lasso_feature_value']
lasso_features['lasso_feature_value'] = lasso_features['lasso_feature_value'].
abs()
lasso_features.sort_values(by=['lasso_feature_value'], ascending = False, inpl
ace = True)
lasso_features.head()
```

Out[73]:

| | lasso_feature | lasso_feature_value |
|---|---|---|
| **106** | x13_ClyTile | 477604.104900 |
| **112** | x13_WdShngl | 36166.251006 |
| **144** | x16_NoRidge | 35413.949849 |
| **151** | x16_StoneBr | 33760.245291 |
| **241** | x29_Ex | 31804.211235 |

In [74]:
```
ridge_features = pd.DataFrame(gcv_ridge.best_estimator_.named_steps['ridge'].c
oef_,np.append (cont_name, preprocess_scaling.fit(x).named_transformers_['Ohe'
].get_feature_names())).reset_index()
ridge_features.columns = ['ridge_feature','ridge_feature_value']
ridge_features['ridge_feature_value'] = ridge_features['ridge_feature_value'].
abs()
ridge_features.sort_values(by=['ridge_feature_value'], ascending = False, inpl
ace = True)
```

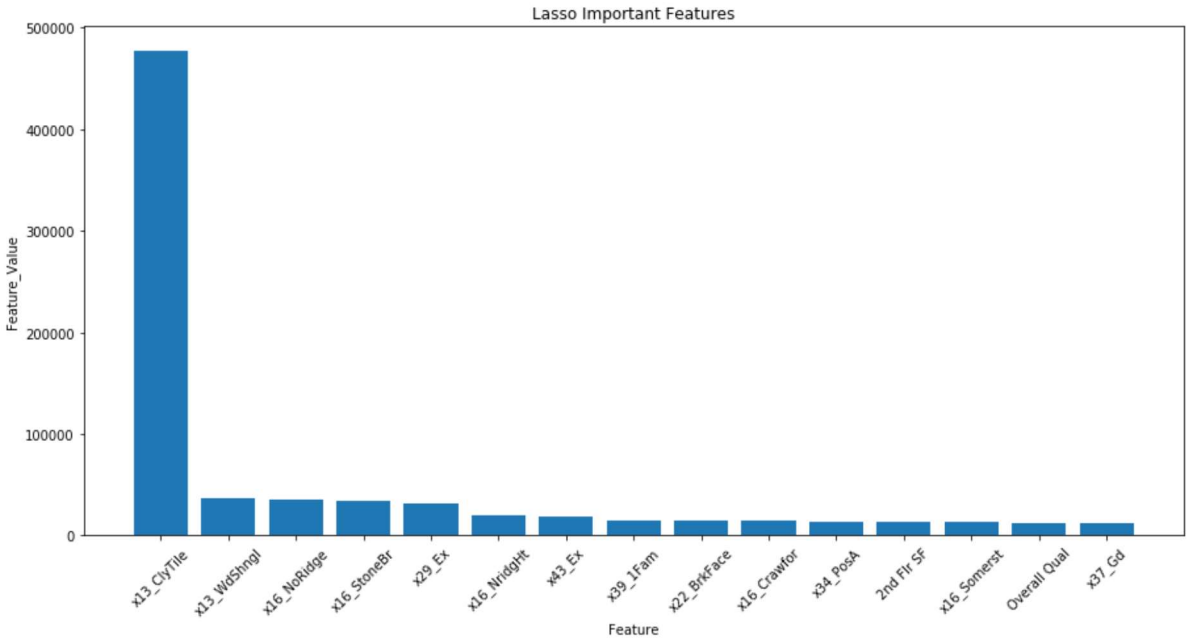
In [75]:
```
elastic_features = pd.DataFrame(gcv_elastic.best_estimator_.named_steps['elast
ic'].coef_,np.append (cont_name, preprocess_scaling.fit(x).named_transformers_
['Ohe'].get_feature_names())).reset_index()
elastic_features.columns = ['elastic_feature','elastic_feature_value']
elastic_features['elastic_feature_value'] = elastic_features['elastic_feature_
value'].abs()
elastic_features.sort_values(by=['elastic_feature_value'], ascending = False,
inplace = True)
```

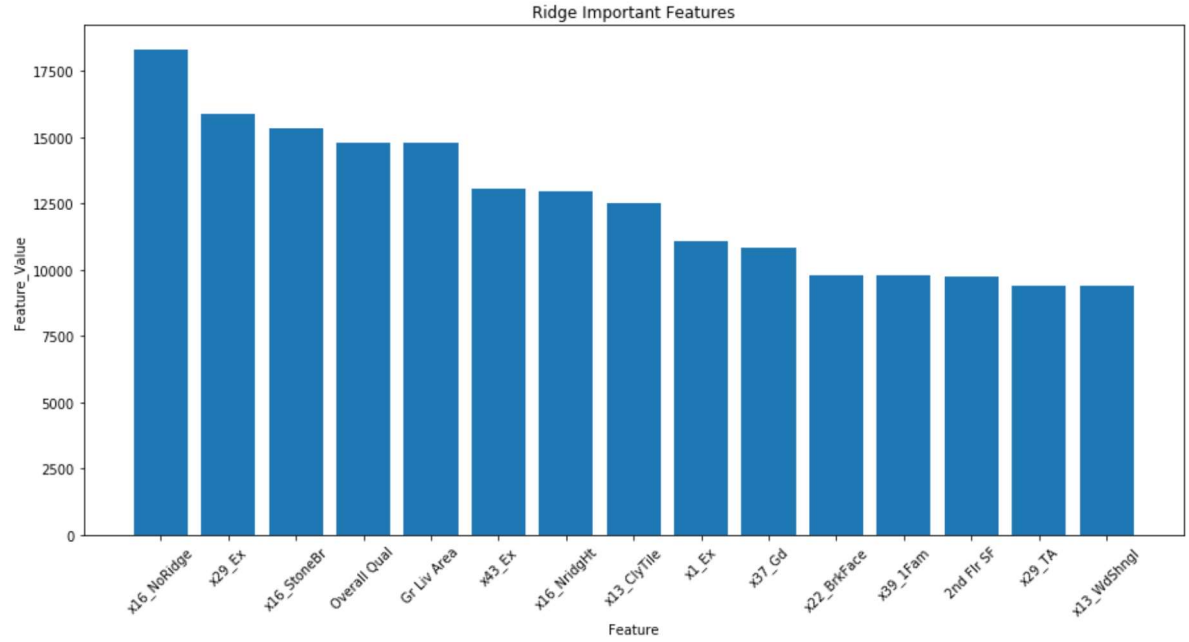
In [76]:
```
lasso_features['lasso_feature'][:15]
```

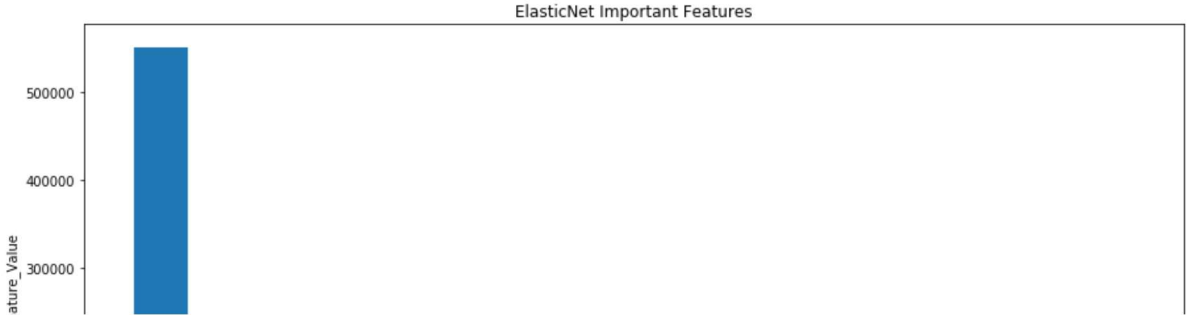
Out[76]:
```
106     x13_ClyTile
112     x13_WdShngl
144     x16_NoRidge
151     x16_StoneBr
241          x29_Ex
145     x16_NridgHt
333          x43_Ex
302        x39_1Fam
183     x22_BrkFace
133     x16_Crawfor
274        x34_PosA
12        2nd Flr SF
150     x16_Somerst
2      Overall Qual
291          x37_Gd
Name: lasso_feature, dtype: object
```

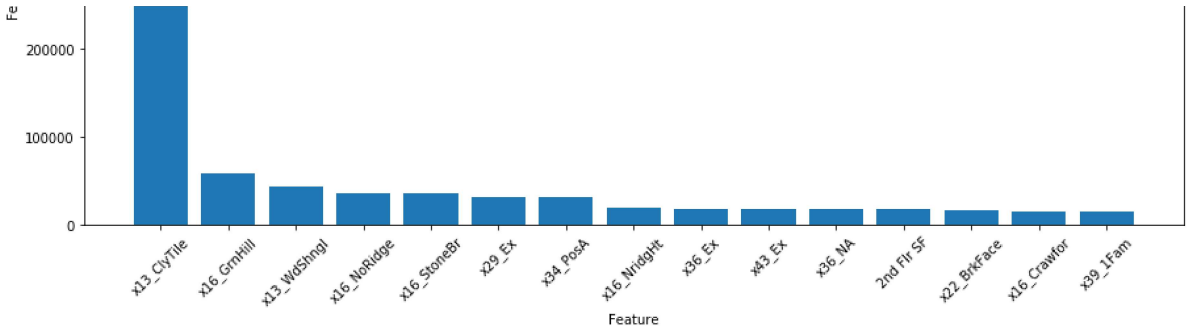
In [77]:
```
fig, ax = plt.subplots(3,1, figsize=(15,30))
plt.subplots_adjust(wspace=0.02,hspace=0.60)

ax[0].bar(lasso_features['lasso_feature'][:15],lasso_features['lasso_feature_v
alue'][:15])
ax[0].set_xticklabels(lasso_features['lasso_feature'][:15], rotation=45)
ax[0].set_title('Lasso Important Features')
ax[0].set_xlabel('Feature')
ax[0].set_ylabel('Feature_Value')

ax[1].bar(ridge_features['ridge_feature'][:15],ridge_features['ridge_feature_v
alue'][:15])
ax[1].set_xticklabels(ridge_features['ridge_feature'][:15], rotation=45)
ax[1].set_title('Ridge Important Features')
ax[1].set_xlabel('Feature')
ax[1].set_ylabel('Feature_Value')

ax[2].bar(elastic_features['elastic_feature'][:15],elastic_features['elastic_f
eature_value'][:15])
ax[2].set_xticklabels(elastic_features['elastic_feature'][:15], rotation=45)
ax[2].set_title('ElasticNet Important Features')
ax[2].set_xlabel('Feature')
ax[2].set_ylabel('Feature_Value')
```

Out[77]: Text(0, 0.5, 'Feature_Value')


Lasso Important Features
Feature_Value
500000
400000
300000
200000
100000
0
x13_ClyTile
x13_WdShngl
x16_NoRidge
x16_StoneBr
x29_Ex
x16_NridgHt
x43_Ex
x39_1Fam
x22_BrkFace
x16_Crawfor
x34_PosA
2nd Flr SF
x16_Somerst
Overall Qual
x37_Gd
Feature


Ridge Important Features
Feature_Value
17500
15000
12500
10000
7500
5000
2500
0
x16_NoRidge
x29_Ex
x16_StoneBr
Overall Qual
Gr Liv Area
x43_Ex
x16_NridgHt
x13_ClyTile
x1_Ex
x37_Gd
x22_BrkFace
x39_1Fam
2nd Flr SF
x29_TA
x13_WdShngl
Feature


ElasticNet Important Features
ature_Value
500000
400000
300000




In [78]:
```
a = list (lasso_features['lasso_feature'][:15])
b= list (ridge_features['ridge_feature'][:15])
c= list (elastic_features['elastic_feature'][:15])

set(a).intersection(b, c)
```

Out[78]:
```
{'2nd Flr SF',
 'x13_ClyTile',
 'x13_WdShngl',
 'x16_NoRidge',
 'x16_NridgHt',
 'x16_StoneBr',
 'x22_BrkFace',
 'x29_Ex',
 'x39_1Fam',
 'x43_Ex'}
```

On running this before submission, 10 out of 15 features are common across these models - from looking at variables, we can further find that certain categorical variables repeat (one-hot encoded forms)

In [ ]:
```

```

In [27]:
```
import pandas as pd
telco_data = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

In [28]:
```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
```

In [30]:
```
telco_data.columns
```

Out[30]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [31]:
```
telco_data.shape
```

Out[31]:
```
(7043, 21)
```

In [32]:
```
telco_data.head()
```

Out[32]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines |
|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No |

5 rows × 21 columns

In [33]:
```
telco_data.dtypes
telco_data.drop(columns= 'customerID', inplace = True)
```

In [34]:
```
telco_data['TotalCharges'] = pd.to_numeric(telco_data['TotalCharges'], errors
= 'coerce')
#converting it to continuous as dtype was object
```

In [35]:
```
telco_continuous = telco_data.select_dtypes (include = ['int64','float64'])
telco_continuous.head()
telco_continuous.drop (columns = ['SeniorCitizen'],inplace = True)
```

In [36]:
```
cont_var = list (telco_continuous)
```

In [37]:
```
cont_var
```

Out[37]:
```
['tenure', 'MonthlyCharges', 'TotalCharges']
```

In [38]:
```
telco_continuous
```

Out[38]:

| | tenure | MonthlyCharges | TotalCharges |
|---|---|---|---|
| **0** | 1 | 29.85 | 29.85 |
| **1** | 34 | 56.95 | 1889.50 |
| **2** | 2 | 53.85 | 108.15 |
| **3** | 45 | 42.30 | 1840.75 |
| **4** | 2 | 70.70 | 151.65 |
| **5** | 8 | 99.65 | 820.50 |
| **6** | 22 | 89.10 | 1949.40 |
| **7** | 10 | 29.75 | 301.90 |
| **8** | 28 | 104.80 | 3046.05 |
| **9** | 62 | 56.15 | 3487.95 |
| **10** | 13 | 49.95 | 587.45 |
| **11** | 16 | 18.95 | 326.80 |
| **12** | 58 | 100.35 | 5681.10 |
| **13** | 49 | 103.70 | 5036.30 |
| **14** | 25 | 105.50 | 2686.05 |
| **15** | 69 | 113.25 | 7895.15 |
| **16** | 52 | 20.65 | 1022.95 |
| **17** | 71 | 106.70 | 7382.25 |
| **18** | 10 | 55.20 | 528.35 |
| **19** | 21 | 90.05 | 1862.90 |
| **20** | 1 | 39.65 | 39.65 |
| **21** | 12 | 19.80 | 202.25 |
| **22** | 1 | 20.15 | 20.15 |
| **23** | 58 | 59.90 | 3505.10 |
| **24** | 49 | 59.60 | 2970.30 |
| **25** | 30 | 55.30 | 1530.60 |
| **26** | 47 | 99.35 | 4749.15 |
| **27** | 1 | 30.20 | 30.20 |
| **28** | 72 | 90.25 | 6369.45 |
| **29** | 17 | 64.70 | 1093.10 |
| **...** | ... | ... | ... |
| **7013** | 40 | 93.40 | 3756.40 |
| **7014** | 41 | 89.20 | 3645.75 |
| **7015** | 34 | 85.20 | 2874.45 |
| **7016** | 1 | 49.95 | 49.95 |

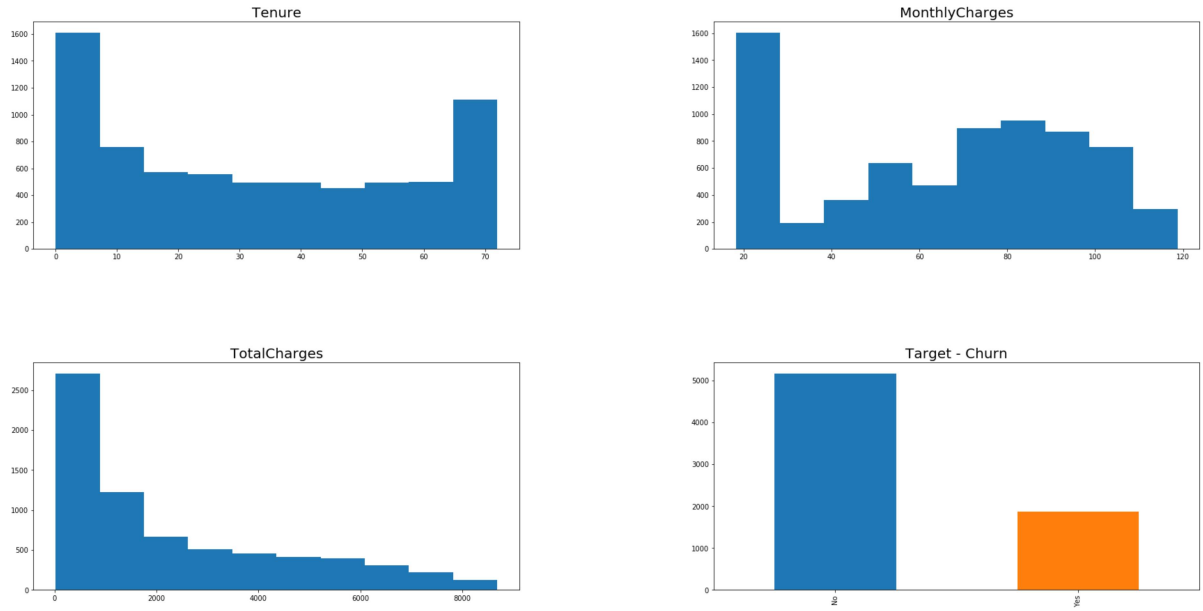
| | tenure | MonthlyCharges | TotalCharges |
|---|---|---|---|
| **7017** | 51 | 20.65 | 1020.75 |
| **7018** | 1 | 70.65 | 70.65 |
| **7019** | 39 | 20.15 | 826.00 |
| **7020** | 12 | 19.20 | 239.00 |
| **7021** | 12 | 59.80 | 727.80 |
| **7022** | 72 | 104.95 | 7544.30 |
| **7023** | 63 | 103.50 | 6479.40 |
| **7024** | 44 | 84.80 | 3626.35 |
| **7025** | 18 | 95.05 | 1679.40 |
| **7026** | 9 | 44.20 | 403.35 |
| **7027** | 13 | 73.35 | 931.55 |
| **7028** | 68 | 64.10 | 4326.25 |
| **7029** | 6 | 44.40 | 263.05 |
| **7030** | 2 | 20.05 | 39.25 |
| **7031** | 55 | 60.00 | 3316.10 |
| **7032** | 1 | 75.75 | 75.75 |
| **7033** | 38 | 69.50 | 2625.25 |
| **7034** | 67 | 102.95 | 6886.25 |
| **7035** | 19 | 78.70 | 1495.10 |
| **7036** | 12 | 60.65 | 743.30 |
| **7037** | 72 | 21.15 | 1419.40 |
| **7038** | 24 | 84.80 | 1990.50 |
| **7039** | 72 | 103.20 | 7362.90 |
| **7040** | 11 | 29.60 | 346.45 |
| **7041** | 4 | 74.40 | 306.60 |
| **7042** | 66 | 105.65 | 6844.50 |

7043 rows × 3 columns

# 2.1

In [39]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(2, 2, figsize=(30, 15))
plt.subplots_adjust(wspace=0.40,hspace=0.50)

ax[0,0].hist(telco_continuous['tenure'])
ax[0,1].hist(telco_continuous['MonthlyCharges'])
ax[1,0].hist(telco_continuous['TotalCharges'])

telco_data['Churn'].value_counts().plot(kind='bar', ax=ax[1,1])

ax[0,0].set_title('Tenure', fontsize = 20)
ax[0,1].set_title('MonthlyCharges', fontsize = 20)
ax[1,0].set_title('TotalCharges', fontsize = 20)
ax[1,1].set_title('Target - Churn', fontsize = 20)
```

Out[39]: Text(0.5, 1.0, 'Target - Churn')




# 2.2

In [40]:
```
cat_var = list(set(telco_data.columns)- set(telco_continuous.columns))
```

In [41]:
```
cat_var
```

Out[41]:
```
['StreamingTV',
 'TechSupport',
 'PaymentMethod',
 'PhoneService',
 'SeniorCitizen',
 'Contract',
 'Dependents',
 'OnlineBackup',
 'Partner',
 'MultipleLines',
 'InternetService',
 'Churn',
 'DeviceProtection',
 'PaperlessBilling',
 'OnlineSecurity',
 'StreamingMovies',
 'gender']
```

In [42]:
```
from sklearn.model_selection import train_test_split
cat_var.remove('Churn')
X_train, X_test, y_train, y_test = train_test_split (telco_data.loc[:, telco_d
ata.columns != 'Churn'],telco_data['Churn'], random_state = 0)
X_train = pd.concat((X_train[cat_var], X_train[cont_var]), axis=1)
```

In [43]:
```
from sklearn.compose import make_column_transformer
from sklearn.compose import ColumnTransformer

from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer


cat_pipe = ColumnTransformer([('imputer', SimpleImputer(strategy = 'median'),
cont_var) , ('ohe', OneHotEncoder(handle_unknown='ignore'), cat_var)], remaind
er='passthrough')
```

In [44]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

logreg_model = make_pipeline(cat_pipe, LogisticRegression())
logreg_scores = cross_val_score(logreg_model, X_train, y_train, cv=10)
print('Logistic Regression: ', np.mean(logreg_scores))


from sklearn.svm import LinearSVC
linearSVC_model = make_pipeline(cat_pipe, LinearSVC())
linearSVC_scores = cross_val_score (linearSVC_model,X_train, y_train, cv=10)
print('Linear SVC: ', np.mean(linearSVC_scores))


from sklearn.neighbors.nearest_centroid import NearestCentroid
ncentroid_model = make_pipeline (cat_pipe,NearestCentroid())
ncentroid_scores = cross_val_score (ncentroid_model, X_train, y_train, cv=10)
print('N Centroid: ', np.mean(ncentroid_scores))
```

```
Logistic Regression:  0.8023444758109355
Linear SVC:  0.7317723821036433
N Centroid:  0.5238651567706784
```

The performance of Logistic Regression is better than Linear SVC and N Centroid

In [45]:
```
from sklearn.preprocessing import StandardScaler
scaler_pipe = ColumnTransformer([('scaling', StandardScaler(), cont_var)], rem
ainder='passthrough')
#cat_con_pipe = ColumnTransformer([('somename', StandardScaler(), ['tenure',
 'MonthlyCharges']),
                                  #('ohe', OneHotEncoder(handle_unknown='ignor
e'), cat_var)], remainder = 'passthrough')
cat_con_pipe = ColumnTransformer([('imputer', SimpleImputer(strategy = 'media
n'), np.arange (0,len(cont_var))) , ('ohe', OneHotEncoder(handle_unknown='igno
re'),  np.arange(len(cont_var),len(X_train.columns)))], remainder='passthroug
h')
```

In [46]:
```
logreg1_model = make_pipeline(scaler_pipe, cat_con_pipe, LogisticRegression())
logreg1_scores = cross_val_score(logreg1_model, X_train, y_train, cv=10)
print('Logistic Regression with Standard Scaling: ', np.mean(logreg1_scores))

linearSVC1_model = make_pipeline(scaler_pipe, cat_con_pipe,LinearSVC())
linearSVC1_scores = cross_val_score (linearSVC1_model,X_train, y_train, cv=10)
print('Linear SVC with Standard Scaling: ', np.mean(linearSVC1_scores))

ncentroid1_model = make_pipeline (scaler_pipe, cat_con_pipe, NearestCentroid
())
ncentroid1_scores = cross_val_score (ncentroid1_model, X_train, y_train, cv=10
)
print('N Centroid with Standard Scaling: ', np.mean(ncentroid1_scores))
```

```
Logistic Regression with Standard Scaling:  0.8025342291316185
Linear SVC with Standard Scaling:  0.8025370987470541
N Centroid with Standard Scaling:  0.7391182389170072
```

We see significant improvement in Linear SVC and N Centroid on Standard Scaling

# 2.3

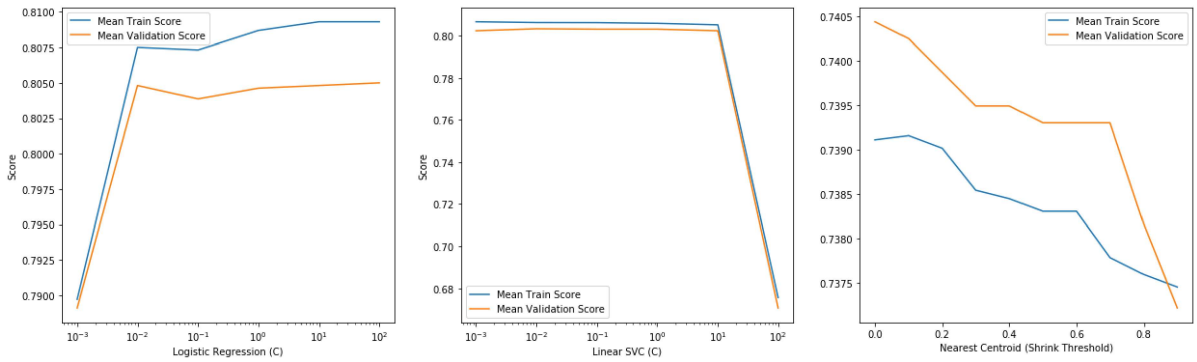
In [47]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline

pipeline_logreg = Pipeline(steps=[('scaling',scaler_pipe), ('preprocess', cat_
con_pipe), ('logreg', LogisticRegression())])
pipeline_linearSVC = Pipeline(steps=[('scaling',scaler_pipe), ('preprocess', c
at_con_pipe), ('linearSVC', LinearSVC())])
pipeline_ncentroid = Pipeline(steps=[('scaling',scaler_pipe), ('preprocess', c
at_con_pipe), ('ncentroid', NearestCentroid())])

paragrid_logreg = {'logreg__C':np.logspace(-3,2,6)}
#'logreg__penalty': ['l1', 'l2']
paragrid_linearSVC = {'linearSVC__C': np.logspace(-3, 2, 6)}
#'linearSVC__gamma': np.logspace(-3, 2, 6) / X_train.shape[0]
paragrid_ncentroid = {'ncentroid__shrink_threshold': np.arange(0,1,0.1)}

gcv_logreg = GridSearchCV(pipeline_logreg, param_grid=paragrid_logreg, cv=5, r
eturn_train_score=True)
gcv_logreg.fit(X_train, y_train)
print("Logistic Regression best parameters with stratified kfold is {}".format
(gcv_logreg.best_params_))
print("Logistic Regression best score with stratified kfold is {}".format(gcv_
logreg.best_score_))


gcv_linearSVC = GridSearchCV(pipeline_linearSVC, param_grid=paragrid_linearSVC
, cv=5, return_train_score=True)
gcv_linearSVC.fit(X_train, y_train)
print("Linear SVC best parameters with stratified kfold is {}".format(gcv_line
arSVC.best_params_))
print("Linear SVC best score with stratified kfold is {}".format(gcv_linearSVC
.best_score_))


gcv_ncentroid = GridSearchCV(pipeline_ncentroid, param_grid=paragrid_ncentroid
, cv=5, return_train_score=True)
gcv_ncentroid.fit(X_train, y_train)
print("Nearest Centroid best parameters with stratified kfold is {}".format(gc
v_ncentroid.best_params_))
print("Nearest Centroid best score with stratified kfold is {}".format(gcv_nce
ntroid.best_score_))
```

```
Logistic Regression best parameters with stratified kfold is {'logreg__C': 10
0.0}
Logistic Regression best score with stratified kfold is 0.8049981067777358
Linear SVC best parameters with stratified kfold is {'linearSVC__C': 0.01}
Linear SVC best score with stratified kfold is 0.8032942067398713
Nearest Centroid best parameters with stratified kfold is {'ncentroid__shrink
_threshold': 0.0}
Nearest Centroid best score with stratified kfold is 0.7404392275653162
```

There is a very slight improvement when compared to the earlier models on using gridsearch

In [48]:
```
fig, ax = plt.subplots(1,3,figsize = (21,6))
logreg_result = pd.DataFrame(gcv_logreg.cv_results_)
ax[0].plot(logreg_result['param_logreg__C'], logreg_result['mean_train_score'
], label = 'Mean Train Score')
ax[0].plot(logreg_result['param_logreg__C'], logreg_result['mean_test_score'],
label = 'Mean Validation Score')
ax[0].set_xscale('log')
ax[0].legend(loc="best")
ax[0].set_xlabel('Logistic Regression (C)')
ax[0].set_ylabel('Score')

linearSVC_result = pd.DataFrame(gcv_linearSVC.cv_results_)
ax[1].plot(linearSVC_result['param_linearSVC__C'], linearSVC_result['mean_trai
n_score'], label = 'Mean Train Score')
ax[1].plot(linearSVC_result['param_linearSVC__C'], linearSVC_result['mean_test
_score'], label = 'Mean Validation Score')
ax[1].set_xscale('log')
ax[1].legend(loc="best")

ax[1].set_xlabel('Linear SVC (C)')
ax[1].set_ylabel('Score')

ncentroid_result = pd.DataFrame(gcv_ncentroid.cv_results_)
ax[2].plot(ncentroid_result['param_ncentroid__shrink_threshold'], ncentroid_re
sult['mean_train_score'], label = 'Mean Train Score')
ax[2].plot(ncentroid_result['param_ncentroid__shrink_threshold'], ncentroid_re
sult['mean_test_score'], label = 'Mean Validation Score')
ax[2].legend(loc="best")

ax[2].set_xlabel('Nearest Centroid (Shrink Threshold)')
ax[1].set_ylabel('Score')
```

Out[48]: Text(0, 0.5, 'Score')




# 2.4

In [53]:
```
from sklearn.model_selection import KFold
kfold = KFold(n_splits=5, shuffle = True)


paragrid_logreg = {'logreg__C':np.logspace(-3,2,6)}
#'logreg__penalty': ['l1', 'l2']
paragrid_linearSVC = {'linearSVC__C': np.logspace(-3, 2, 6)}
#'linearSVC__gamma': np.logspace(-3, 2, 6) / X_train.shape[0]
paragrid_ncentroid = {'ncentroid__shrink_threshold': np.arange(0,1,0.1)}

gcv_logreg_kfold = GridSearchCV(pipeline_logreg, param_grid=paragrid_logreg, c
v=kfold, return_train_score=True)
gcv_logreg_kfold.fit(X_train, y_train)
print("Logistic Regression best parameter with kfold is {}".format(gcv_logreg_
kfold.best_params_))
print("Logistic Regression best score with kfold is {}".format(gcv_logreg_kfol
d.best_score_))


gcv_linearSVC_kfold = GridSearchCV(pipeline_linearSVC, param_grid=paragrid_lin
earSVC, cv=kfold, return_train_score=True)
gcv_linearSVC_kfold.fit(X_train, y_train)
print("Linear SVC best parameter with kfold is {}".format(gcv_linearSVC_kfold.
best_params_))
print("Linear SVC best score with kfold is {}".format(gcv_linearSVC_kfold.best
_score_))


gcv_ncentroid_kfold = GridSearchCV(pipeline_ncentroid, param_grid=paragrid_nce
ntroid, cv=kfold, return_train_score=True)
gcv_ncentroid_kfold.fit(X_train, y_train)
print("Nearest Centroid best parameter with kfold is {}".format(gcv_ncentroid_
kfold.best_params_))
print("Nearest Centroid best score with kfold is {}".format(gcv_ncentroid_kfol
d.best_score_))
```

```
Logistic Regression best parameter with kfold is {'logreg__C': 10.0}
Logistic Regression best score with kfold is 0.8068913290420295
Linear SVC best parameter with kfold is {'linearSVC__C': 0.1}
Linear SVC best score with kfold is 0.8059447179098826
Nearest Centroid best parameter with kfold is {'ncentroid__shrink_threshold':
0.30000000000000004}
Nearest Centroid best score with kfold is 0.7387353275274517
```

- C parameter for logistic regression is constant from 100 to 10
- C parameter for linear SVC changed from 0.01 to 0.001
- C parameter for N Centroid changes from 0.0 to 0.4

In [51]:
```
kfold = KFold(n_splits=5, shuffle = True,random_state = 1)

gcv_logreg_kfold_seed= GridSearchCV(pipeline_logreg, param_grid=paragrid_logre
g, cv=kfold, return_train_score=True)
gcv_logreg_kfold_seed.fit(X_train, y_train)
print("Logistic Regression best parameter with shuffle and seed is {}".format(
gcv_logreg_kfold_seed.best_params_))
print("Logistic Regression best score with shuffle and seed is {}".format(gcv_
logreg_kfold_seed.best_score_))


gcv_linearSVC_kfold_seed = GridSearchCV(pipeline_linearSVC, param_grid=paragri
d_linearSVC, cv=kfold, return_train_score=True)
gcv_linearSVC_kfold_seed.fit(X_train, y_train)
print("Linear SVC best parameter with shuffle and seed is {}".format(gcv_linea
rSVC_kfold_seed.best_params_))
print("Linear SVC best score with shuffle and seed is {}".format(gcv_linearSVC
_kfold_seed.best_score_))


gcv_ncentroid_kfold_seed = GridSearchCV(pipeline_ncentroid, param_grid=paragri
d_ncentroid, cv=kfold, return_train_score=True)
gcv_ncentroid_kfold_seed.fit(X_train, y_train)
print("Nearest Centroid best parameter with shuffle and seed is {}".format(gcv
_ncentroid_kfold_seed.best_params_))
print("Nearest Centroid best score with shuffle and seed is {}".format(gcv_nce
ntroid_kfold_seed.best_score_))
```

```
Logistic Regression best parameter with shuffle and seed is {'logreg__C': 0.0
1}
Logistic Regression best score with shuffle and seed is 0.8065126845891708
Linear SVC best parameter with shuffle and seed is {'linearSVC__C': 0.001}
Linear SVC best score with shuffle and seed is 0.8055660734570238
Nearest Centroid best parameter with shuffle and seed is {'ncentroid__shrink_
threshold': 0.4}
Nearest Centroid best score with shuffle and seed is 0.7391139719803105
```

The below parameters change in comparison to 2.3 Stratified K-fold

- C parameter for logistic regression changed from 100 to 0.01
- C parameter for linear SVC changed from 0.01 to 0.001
- C parameter for N Centroid changed from 0 to 0.4

In [54]:
```
X1_train, X1_test, y1_train, y1_test = train_test_split (telco_data.loc[:, tel
co_data.columns != 'Churn'],telco_data['Churn'], random_state = 10)
X1_train = pd.concat((X1_train[cat_var], X1_train[cont_var]), axis=1)


gcv_logreg_kfold_tt = GridSearchCV(pipeline_logreg, param_grid=paragrid_logreg
, cv=kfold, return_train_score=True)
gcv_logreg_kfold_tt.fit(X1_train, y1_train)
print("Logistic Regression best parameter with kfold and train test seed chang
e is {}".format(gcv_logreg_kfold_tt.best_params_))
print("Logistic Regression best score with kfold and train test seed change is
{}".format(gcv_logreg_kfold_tt.best_score_))


gcv_linearSVC_kfold_tt = GridSearchCV(pipeline_linearSVC, param_grid=paragrid_
linearSVC, cv=kfold, return_train_score=True)
gcv_linearSVC_kfold_tt.fit(X1_train, y1_train)
print("Linear SVC best parameter with kfold and train test seed change is {}".
format(gcv_linearSVC_kfold_tt.best_params_))
print("Linear SVC best score with kfold and train test seed change is {}".form
at(gcv_linearSVC_kfold_tt.best_score_))


gcv_ncentroid_kfold_tt = GridSearchCV(pipeline_ncentroid, param_grid=paragrid_
ncentroid, cv=kfold, return_train_score=True)
gcv_ncentroid_kfold_tt.fit(X1_train, y1_train)
print("Nearest Centroid best parameter with kfold and train test seed change i
s {}".format(gcv_ncentroid_kfold_tt.best_params_))
print("Nearest Centroid best score with kfold and train test seed change is {}
".format(gcv_ncentroid_kfold_tt.best_score_))
```

```
Logistic Regression best parameter with kfold and train test seed change is
{'logreg__C': 0.01}
Logistic Regression best score with kfold and train test seed change is 0.805
1874290041651
Linear SVC best parameter with kfold and train test seed change is {'linearSV
C__C': 0.1}
Linear SVC best score with kfold and train test seed change is 0.804240817872
0182
Nearest Centroid best parameter with kfold and train test seed change is {'nc
entroid__shrink_threshold': 0.0}
Nearest Centroid best score with kfold and train test seed change is 0.738356
683074593
```

The below parameters change in comparison to 2.3 Stratified K-fold

- C parameter for logistic regression changed from 100 to 0.01
- C parameter for linear SVC changed from 0.01 to 0.001
- C parameter for N Centroid is constant from at 0.0

# 2.5

# Logistic Regression best score with kfold --> C = 1 and score = 0.8068

# Linear SVC best score also with kfold --> c = 10 and score = 0.8059

In [57]:
```
coeff_logreg = gcv_logreg_kfold.best_estimator_.named_steps['logreg'].coef_[0]
coeff_linearSVC = gcv_linearSVC_kfold.best_estimator_.named_steps['linearSVC']
.coef_[0]
```

In [77]:
```
len(coeff_logreg)
```

Out[77]: 46

In [83]:
```
fig, ax = plt.subplots(1,2, figsize=(16,6))
plt.subplots_adjust(wspace=0.2,hspace=0.60)

ax[0].plot(coeff_logreg, 'o', label = "C = 10.0", color = 'red')
ax[0].set_title ('Co-efficients of Best Performing Logistic Regression (k-fol
d)')
ax[0].set_ylabel ('Co-efficient Value')
ax[0].legend()
ax[0].set_xlabel ('Feature Position')



ax[1].plot(coeff_linearSVC, 'o', label = "C = 0.1")
ax[1].set_title ('Co-efficients of Best Performing Linear SVC (k-fold)')
ax[1].set_ylabel ('Co-efficient Value')
ax[1].legend()
ax[1].set_xlabel ('Feature Position')
```

Out[83]: Text(0.5, 0, 'Feature Position')