SUMMARY

DataFrame dfa -

Data with dummy variables for Location, Category, Worktype and Job Class (Data Scintist. Data Analyst, Business Intelligence). Training dataset was created with rows where FinalSal is NOT NULL. Test dataset was created with rows where FinalSal IS NULL. MLR OLS regression was run for three scenarios

1. Features = Location, Job Class, Category, Work Type MAE 44301, MSA 6014002312, R2 0.044
2. Features = Location, Job Class, Work Type MAE 40480, MSA 5322109879, R2 0.154 >> best and overall best
3. Features = Location, Job Class MAE 45156, MSA 6220574795, R2 0.011 Visual inspection of the plots also indicate that Scenario 2 is the best.

DataFrame dfn -

Data with numeric values mapped for all classifier datapoints. No dummy variables created. Training dataset was created with rows where FinalSal is NOT NULL. Test dataset was created with rows where FinalSal IS NULL.

KNN regression was run for three scenarios Accuracy scores were taken for Odd number of neighbours between 1 and 51.

1. Features = Location, Job Class, Category, Work Type MAE 44152, MSA 6005511961, R2 0.045 Neighbors 21 >> best
2. Features = Location, Job Class, Work Type MAE 44568, MSA 6223339862, R2 0.010 Neighbors 27
3. Features = Location, Job Class MAE 44568, MSA 6223339862. R2 0.011 Neighbors 27

CART regression was run for three scenarios Accuracy scores were taken for Depth between 1 and 7 and None.

1. Features = Location, Job Class, Category, Work Type MAE 43354, MSA 5768771897, R2 0.083 Depth 2 >> best
2. Features = Location, Job Class, Work Type MAE 44867, MSA 6159154547, R2 0.020 Depth 4
3. Features = Location, Job Class MAE 44867, MSA 6159154547, R2 0.021 Depth 4

Enhancements if there is time

1. Use bootstrapping to increase size of Train database and check accuracy of OLS model

In [4]:

```
import requests
from bs4 import BeautifulSoup as bs
import pandas as pd
import numpy as np
import csv
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

LOAD DATA

In [5]:

```
# This is data where dummy variables have been created.
dfe = pd.read_csv('./Data/dfe.csv')
dfe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1286 entries, 0 to 1285
Data columns (total 21 columns):
Unnamed: 0      1286 non-null int64
jobid           1286 non-null int64
finalsal        279 non-null float64
category_HR     1286 non-null int64
category_IT     1286 non-null int64
category_MC     1286 non-null int64
category_SAL    1286 non-null int64
category_ST     1286 non-null int64
location_ADE    1286 non-null int64
location_BRI    1286 non-null int64
location_HOB    1286 non-null int64
location_MEL    1286 non-null int64
location_PER    1286 non-null int64
location_SYD    1286 non-null int64
jobclass_BI     1286 non-null int64
jobclass_DA     1286 non-null int64
jobclass_DS     1286 non-null int64
worktype_CAS    1286 non-null int64
worktype_CON    1286 non-null int64
worktype_FT     1286 non-null int64
worktype_PT     1286 non-null int64
dtypes: float64(1), int64(20)
memory usage: 211.1 KB
```

In [6]:

```
# This is data where salaries are not null and dummy variables have been created.
train = dfe[dfe.finalsal.notnull()]
# This is data where salaries are null and dummy variables have been created.
test = dfe[dfe.finalsal.isnull()]
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1007 entries, 1 to 1284
Data columns (total 21 columns):
Unnamed: 0      1007 non-null int64
jobid           1007 non-null int64
finalsal        0 non-null float64
category_HR     1007 non-null int64
category_IT     1007 non-null int64
category_MC     1007 non-null int64
category_SAL    1007 non-null int64
category_ST     1007 non-null int64
location_ADE    1007 non-null int64
location_BRI    1007 non-null int64
location_HOB    1007 non-null int64
location_MEL    1007 non-null int64
location_PER    1007 non-null int64
location_SYD    1007 non-null int64
jobclass_BI     1007 non-null int64
jobclass_DA     1007 non-null int64
jobclass_DS     1007 non-null int64
worktype_CAS    1007 non-null int64
worktype_CON    1007 non-null int64
worktype_FT     1007 non-null int64
worktype_PT     1007 non-null int64
dtypes: float64(1), int64(20)
memory usage: 173.1 KB
```

Multivariate Linear Regression

In [8]:

```
import statsmodels.api as sm
```

```
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\statsmodels\compat\panda
s.py:56: FutureWarning: The pandas.core.datetools module is deprecated and w
ill be removed in a future version. Please use the pandas.tseries module ins
tead.
  from pandas.core import datetools
```

In [9]:

```
# Setup features for modelling with Ordinary Least Squares
X = train[['category_IT','category_MC','category_SAL','category_ST',\
          'location_BRI','location_HOB','location_MEL','location_PER','location_SYD',\
          'jobclass_BI','jobclass_DA','jobclass_DS','worktype_CON','worktype_FT','worktype_
Y = train['finalsal']
mlrmod1 = sm.OLS(Y,X).fit()
y_pred = mlrmod1.predict(X)
```

In [123]:

```
print('MAE', mean_absolute_error(Y, y_pred))
print('MSA', mean_squared_error(Y, y_pred))
print('R2', r2_score(Y, y_pred))
```

```
MAE 44300.8903268761
MSA 6014002312.054384
R2 0.04403437951642675
```

In [11]:

```
# Get errors / residuals and predicted into dataframe
train['residual'] = Y -y_pred
train['pred'] = y_pred
```

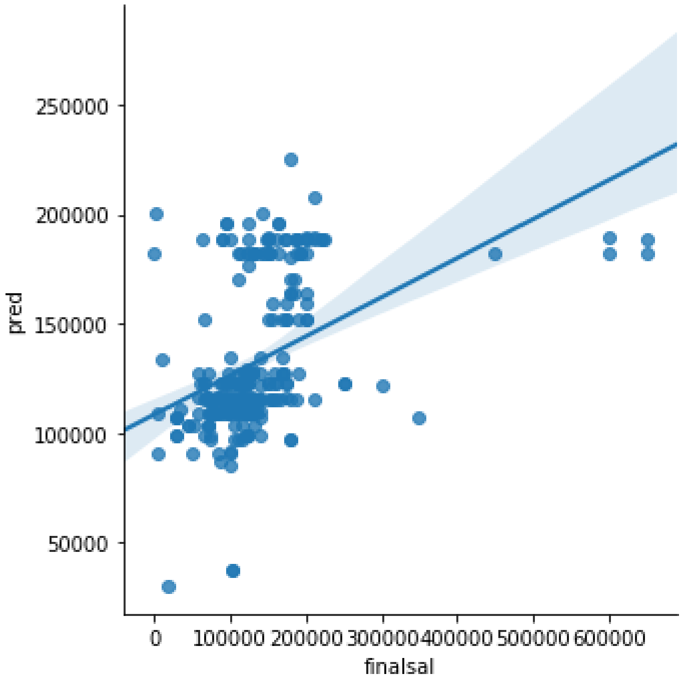
```
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:2:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
```

In [12]:

```
# Plot actual salary and predicted salary
sns.lmplot(x='finalsal', y='pred', data=train)
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x2b501afb588>
```

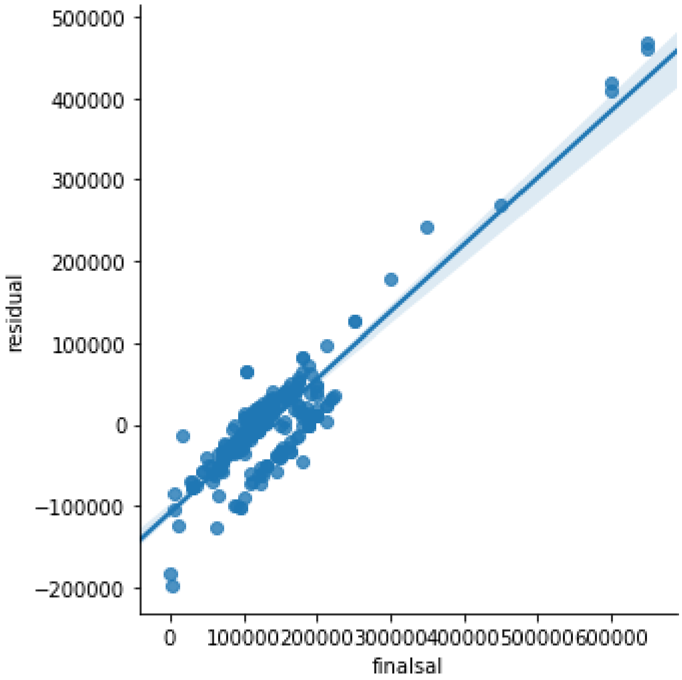


In [13]:

```
# Plot actual salary and residuals
sns.lmplot(x='finalsal', y='residual', data=train)
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x2b501c07588>
```



In [14]:

```
# Remove category from features
X1 = train[['location_BRI','location_HOB','location_MEL','location_PER','location_SYD',\
          'jobclass_BI','jobclass_DA','jobclass_DS','worktype_CON','worktype_FT','worktype_
Y1 = train['finalsal']
mlrmod2 = sm.OLS(Y1,X1).fit()
y_pred1 = mlrmod2.predict(X1)
```

In [15]:

```
print('MAE', mean_absolute_error(Y1, y_pred1))
print('MSA', mean_squared_error(Y1, y_pred1))
print('R2', r2_score(Y1, y_pred1))
```

```
MAE 40480.24195767891
MSA 5322109878.8477125
R2 0.15401527824214778
```

In [17]:

```
train['residual1'] = Y1 -y_pred1
train['pred1'] = y_pred1
```

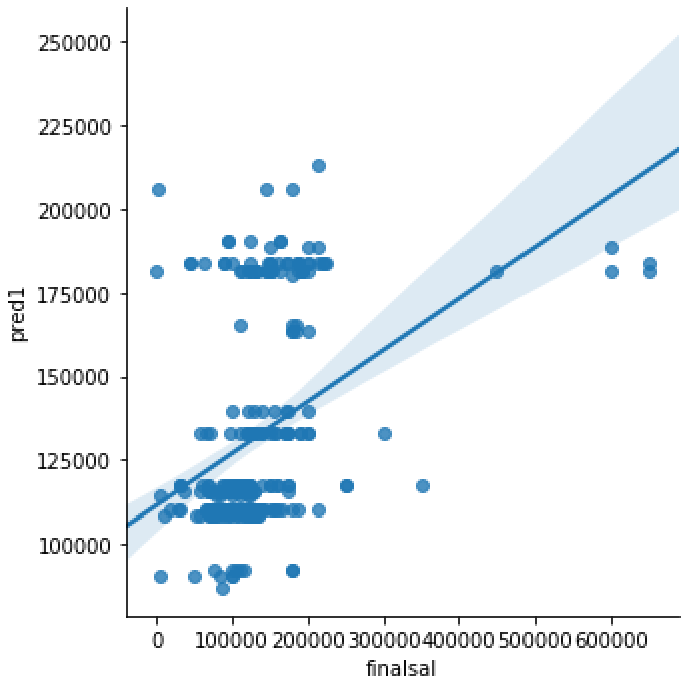
```
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  This is separate from the ipykernel package so we can avoid doing imports
 until
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
```

In [19]:

```
# Linearity plot
sns.lmplot(x='finalsal', y='pred1', data=train)
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x2b502d1fb38>
```

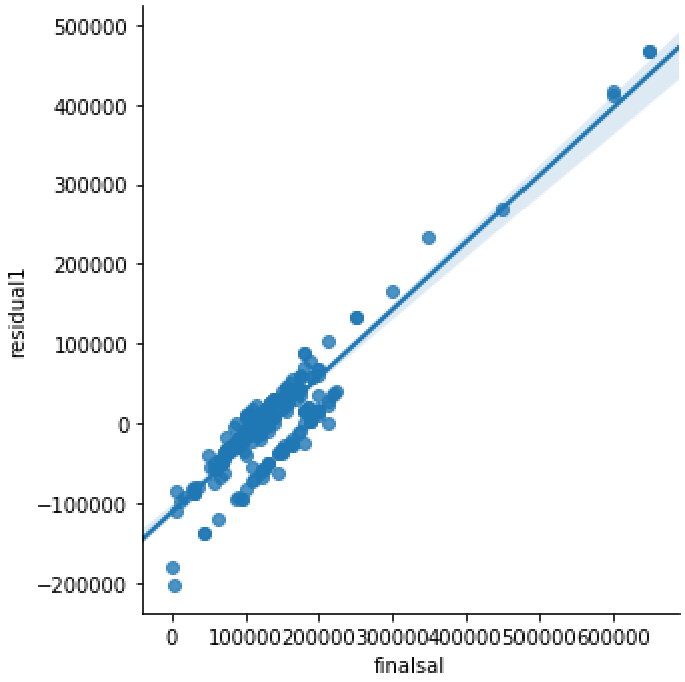


In [20]:

```
# Equality of variance.
sns.lmplot(x='finalsal', y='residual1', data=train)
```

Out[20]:

```
<seaborn.axisgrid.FacetGrid at 0x2b502d851d0>
```

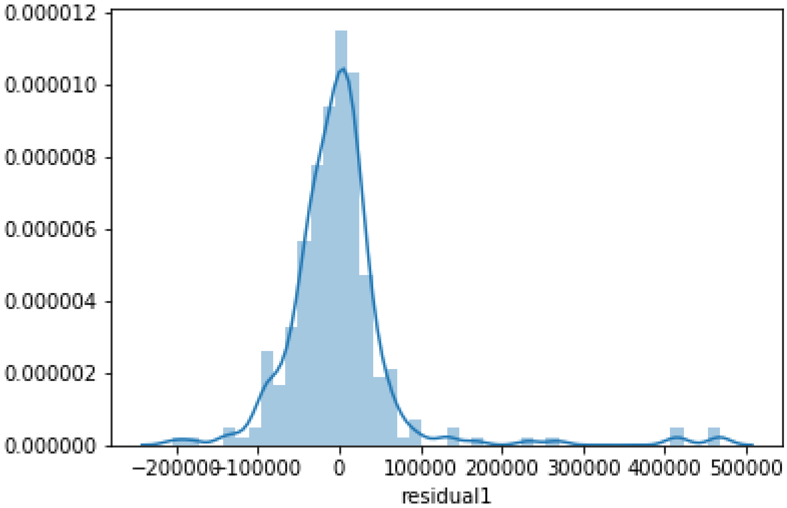


In [21]:

```
# Normality of errors
sns.distplot(train['residual1'])
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2b502dc00b8>
```



In [22]:

```
# Remove worktype from features
X2 = train[['location_BRI','location_HOB','location_MEL','location_PER','location_SYD',\
          'jobclass_BI','jobclass_DA','jobclass_DS']]
Y2 = train['finalsal']
mlrmod3 = sm.OLS(Y2,X2).fit()
y_pred2 = mlrmod3.predict(X2)
```

In [23]:

```
print('MAE', mean_absolute_error(Y2, y_pred2))
print('MSA', mean_squared_error(Y2, y_pred2))
print('R2', r2_score(Y2, y_pred2))
```

```
MAE 45155.65857510886
MSA 6220574795.513222
R2 0.01119831094213597
```

KNN Classification

In [24]:

```
# Load data with numeric values mapped for all classifier datapoints. No dummy variables cr
dfn = pd.read_csv('./Data/dfn.csv')
```

In [25]:

```
dfn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1286 entries, 0 to 1285
Data columns (total 7 columns):
Unnamed: 0    1286 non-null int64
jobid         1286 non-null int64
category      1286 non-null int64
location      1286 non-null int64
jobclass      1286 non-null int64
finalsal      279 non-null float64
worktype      1286 non-null int64
dtypes: float64(1), int64(6)
memory usage: 70.4 KB
```

In [26]:

```
ktrain = dfn[dfn.finalsal.notnull()]
ktest = dfn[dfn.finalsal.isnull()]
```

In [27]:

```
# Loading library
from sklearn.cross_validation import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier,KNeighborsRegressor
from sklearn.metrics import accuracy_score
```

```
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\sklearn\cross_validatio
n.py:41: DeprecationWarning: This module was deprecated in version 0.18 in f
avor of the model_selection module into which all the refactored classes and
functions are moved. Also note that the interface of the new CV iterators ar
e different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [30]:

```
kX = ktrain[['category','location','jobclass','worktype']]
kY = ktrain['finalsal']
ktX = ktest[['category','location','jobclass']]
ktY = ktest['finalsal']
```

In [31]:

```
# Check for model accuracy
sc=[]
for k in range(1,51,2):
    x=[]
    knn = KNeighborsRegressor(n_neighbors=k)
    knn.fit(kX,kY)
    x=[k,knn.score(kX,kY)]
    sc.append(x)
```

In [43]:

```
# Neighbours value of 21 gives the best accuracy
knn = KNeighborsRegressor(n_neighbors=21)
knn.fit(kX,kY)
predict = knn.predict(kX)
```

In [45]:

```
print('MAE', mean_absolute_error(kY, predict))
print('MSA', mean_squared_error(kY, predict))
print('R2', r2_score(kY, predict))
```

```
MAE 44152.29527223075
MSA 6005511961.117378
R2 0.04538397720203946
```

In [58]:

```
kX1 = ktrain[['location','jobclass','worktype']]
kY1 = ktrain['finalsal']
sc=[]
for k in range(1,51,2):
    x=[]
    knn1 = KNeighborsRegressor(n_neighbors=k)
    knn1.fit(kX1,kY1)
    x=[k,knn1.score(kX1,kY1)]
    sc.append(x)
#sc
```

In [59]:

```
knn1 = KNeighborsRegressor(n_neighbors=27)
knn1.fit(kX1,kY1)
predict1 = knn1.predict(kX1)
```

In [60]:

```
print('MAE', mean_absolute_error(kY1, predict1))
print('MSA', mean_squared_error(kY1, predict1))
print('R2', r2_score(kY1, predict1))
```

```
MAE 44568.549449090664
MSA 6223339861.900076
R2 0.010758785270706173
```

In [62]:

```
kX2 = ktrain[['location','jobclass']]
kY2 = ktrain['finalsal']
sc=[]
for k in range(1,51,2):
    x=[]
    knn2 = KNeighborsRegressor(n_neighbors=k)
    knn2.fit(kX2,kY2)
    x=[k,knn2.score(kX2,kY2)]
    sc.append(x)
#sc
```

In [63]:

```
#  [27, 0.010758785270706173],
knn2 = KNeighborsRegressor(n_neighbors=27)
knn2.fit(kX2,kY2)
predict2 = knn2.predict(kX2)
```

In [64]:

```
print('MAE', mean_absolute_error(kY2, predict2))
print('MSA', mean_squared_error(kY2, predict2))
print('R2', r2_score(kY2, predict2))
```

```
MAE 44568.549449090664
MSA 6223339861.900076
R2 0.010758785270706173
```

In [104]:

```
ktrain['predict'] = predict
ktrain['knndiff'] = ktrain['finalsal'] - ktrain['predict']
```

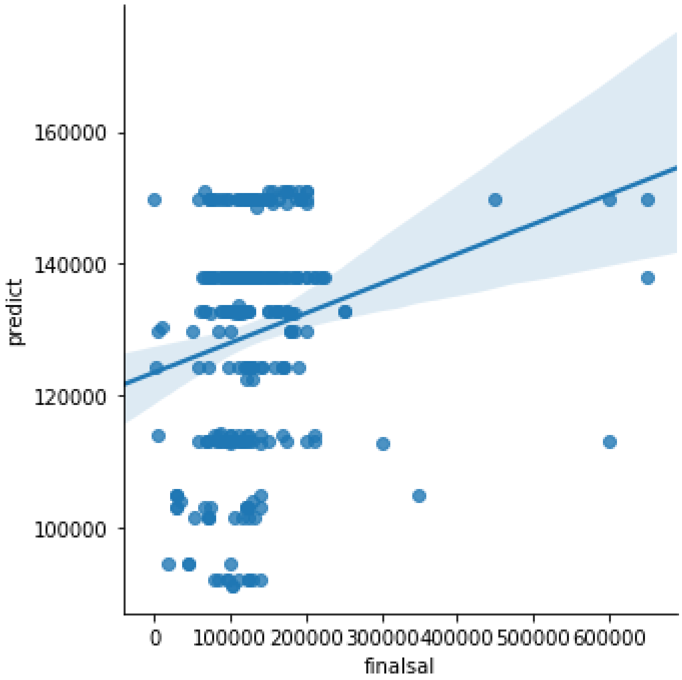
```
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:2:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
```

In [102]:

```
sns.lmplot(x='finalsal', y='predict', data=ktrain)
```

Out[102]:

```
<seaborn.axisgrid.FacetGrid at 0x1d9b838d518>
```

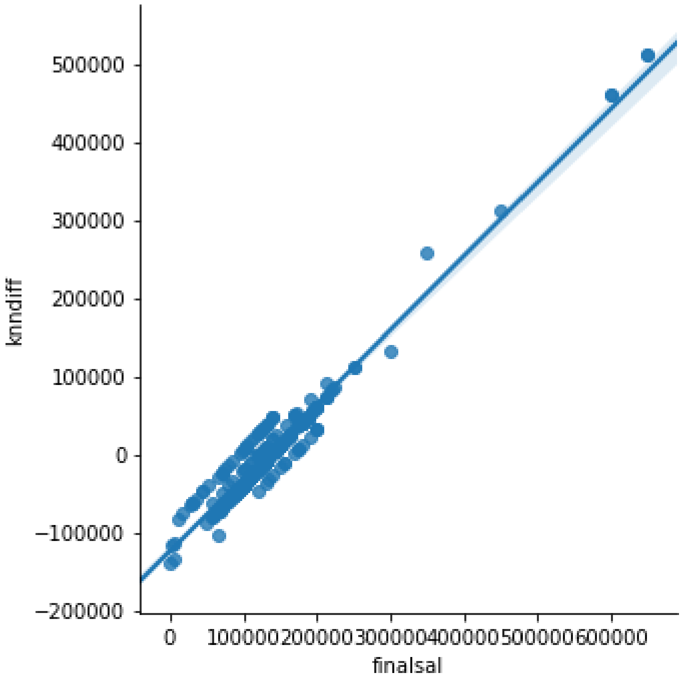


In [105]:

```
sns.lmplot(x='finalsal', y='knndiff', data=ktrain)
```

Out[105]:

```
<seaborn.axisgrid.FacetGrid at 0x1d9b8e494e0>
```



CART Regression Decision Tree

In [66]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [67]:

```
# kX and kY variables
# Get the best depth
sc=[]
for d in [1,2,3,4,5,6,7,None]:
    dtr = DecisionTreeRegressor(max_depth=d)
    dtr.fit(kX, kY)
    dtr_scores = cross_val_score(dtr, kX, kY, cv=3)
    print(d, np.mean(dtr_scores))
```

```
1 -0.041955806818952356
2 -0.02580911105425221
3 -0.037132341316682194
4 -0.07000370264515099
5 -0.05102493464926883
6 -0.051050275369455576
7 -0.05930228972147781
None -0.0707021316633425
```

In [72]:

```
# Depth 2 is the best score
dtr = DecisionTreeRegressor(max_depth=4)
dtr.fit(kX, kY)
predictcart = dtr.predict(kX)
```

In [73]:

```
print('MAE', mean_absolute_error(kY, predictcart))
print('MSA', mean_squared_error(kY, predictcart))
print('R2', r2_score(kY, predictcart))
```

```
MAE 43353.78962406638
MSA 5768771897.200209
R2 0.08301538310327672
```

In [78]:

```
sc=[]
for d in [1,2,3,4,5,6,7,None]:
    dtr1 = DecisionTreeRegressor(max_depth=d)
    dtr1.fit(kX1, kY1)
    dtr_scores = cross_val_score(dtr1, kX1, kY1, cv=3)
    print(d, np.mean(dtr_scores))
```

```
1 -0.028327693066799275
2 -0.02975594655907791
3 -0.05755715832111874
4 -0.057774538046887626
5 -0.057774538046887626
6 -0.057774538046887626
7 -0.057774538046887626
None -0.057774538046887626
```

In [80]:

```
# Depth 4 is the best score
dtr1 = DecisionTreeRegressor(max_depth=4)
dtr1.fit(kX1, kY1)
predictcart1 = dtr1.predict(kX1)
```

In [81]:

```
print('MAE', mean_absolute_error(kY1, predictcart1))
print('MSA', mean_squared_error(kY1, predictcart1))
print('R2', r2_score(kY1, predictcart1))
```

```
MAE 44866.632749691715
MSA 6159154547.129303
R2 0.020961467457540017
```

In [82]:

```
sc=[]
for d in [1,2,3,4,5,6,7,None]:
    dtr2 = DecisionTreeRegressor(max_depth=d)
    dtr2.fit(kX2, kY2)
    dtr_scores = cross_val_score(dtr2, kX2, kY2, cv=3)
    print(d, np.mean(dtr_scores))
```

```
1 -0.028327693066799275
2 -0.02975594655907791
3 -0.05755715832111874
4 -0.057774538046887626
5 -0.057774538046887626
6 -0.057774538046887626
7 -0.057774538046887626
None -0.057774538046887626
```

In [83]:

```
# Depth 4 is the best score
dtr2 = DecisionTreeRegressor(max_depth=4)
dtr2.fit(kX2, kY2)
predictcart2 = dtr2.predict(kX2)
```

In [84]:

```
print('MAE', mean_absolute_error(kY2, predictcart2))
print('MSA', mean_squared_error(kY2, predictcart2))
print('R2', r2_score(kY2, predictcart2))
```

```
MAE 44866.632749691715
MSA 6159154547.129303
R2 0.020961467457540017
```

In [106]:

```
ktrain['predictcart'] = predictcart
ktrain['cartdiff'] = ktrain['finalsal'] - ktrain['predictcart']
```

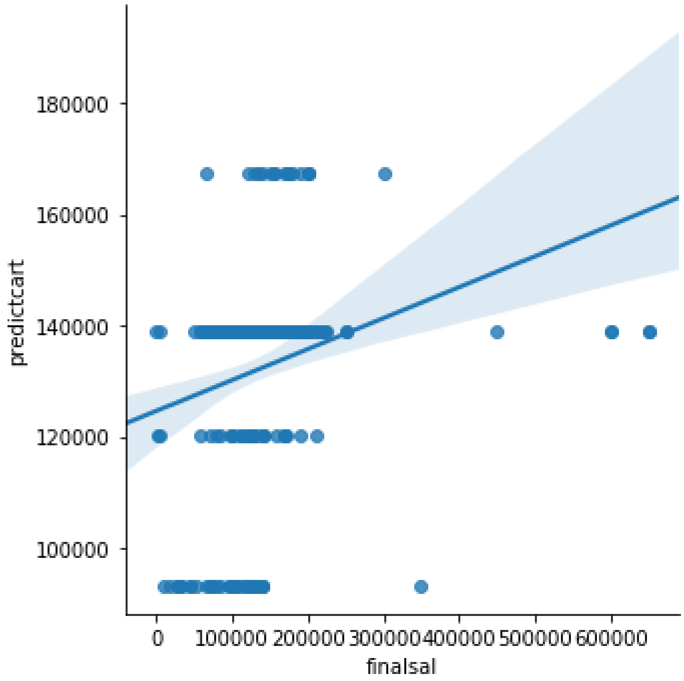
```
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\ipykernel_launcher.py:2:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
```

In [103]:

```
sns.lmplot(x='finalsal', y='predictcart', data=ktrain)
```

Out[103]:

```
<seaborn.axisgrid.FacetGrid at 0x1d9b8dddef0>
```

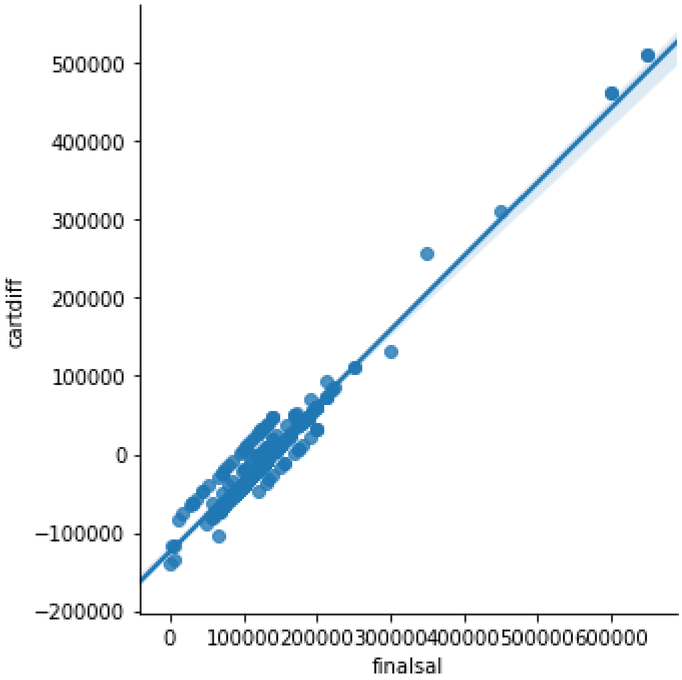


In [107]:

```
sns.lmplot(x='finalsal', y='cartdiff', data=ktrain)
```

Out[107]:

```
<seaborn.axisgrid.FacetGrid at 0x1d9b8e96390>
```



In [ ]:

```
```