Cogs109_FinalProject_Team25

December 15, 2020

1 Cogs 109: Final Project Team 25

```
[24]: from IPython.display import Image
```

1.1 Team Members & Majors

Names	Year	Major
Adrianna Hohil	4th	Cognitive Science Spec. Machine Learning
Bora Mutluoglu	4th	Cognitive Science Spec. Machine Learning
Errick Santos	4th	Data Science
William Hwong	4th	Computer Science

1.2 Overview

In this project our team will be exploring a publicly available survey data set from Stack Overflow focusing on responses from the 2019 and 2020 results. The team intends to perform multivariate linear regression to better understand if there is a relationship between a software programmer's reported job satisfaction, the hours per week they report working, and the overall compensation they've listed. Cross validation methods will be utilized in conjunction with the multivariate regression in an effort to have a more concrete conclusion to our analysis of the models we develop. Due to the voluntary nature of the dataset and potential unreliability there may be some assumptions taken by the project team although it will be clearly noted and explained where intentional manipulations of the data set were made.

1.3 Research Question

This project's research will primarily focus on analyzing if there exists a relationship between the reported job satisfaction of a software programmer relative to the amount of hours worked in a week and their reported compensation (standardized to U.S. Dollars).

1.4 Background & Prior Work

It is no secret that in the past two decades with the rise of technology giants such as Facebook, Netflix, and Apple the computer science field has become immensely popular and lucrative. Software programmers in the United States possessed a median salary of around 84,000 U.S Dollars with the bottom quartile of the field still bringing close to 65,000 U.S Dollars home annually in 2018 (1). If an individual is successful in obtaining a programming role at one of the top companies such as Google or Facebook he or she can expect to lounge in futuristic offices with all inclusive amenities such as: chef staffed kitchens for personally cooked meals, a private barbershop, or even massage parlors to help relieve stress and relax the workers. These tantalizing amenities are only just a bonus to the hefty salaries and stock options that the programmers are often compensated with. From an outsider's perspective it seems impossible for software developers to be dissatisfied with their jobs and high pay even if it isn't at a renowned company like Google.

What many outsiders do not understand is that not all software developers are created equal. In fact the very term "programmer" or "developer" is a very broad term that encompasses a wide range of specialities such as systems engineers, iOS app developers, or video game developers. The video game industry has long been plagued with rumours of overworked workers especially during "crunch time" or the period when a major game release is expected. A senior executive at RockStar Games, publisher of the beloved game Red Dead Redemption 2 openly admitted that it had employees work 100 hours a week to prepare the game for release (2). It's important to note that within the video game industry it is seen as a badge of honor to be able to participate in the development of beloved games and that many major studios have had similar situations of having long work weeks to the detriment of the developer.

While the video game industry might possess a more unique affliction of grueling work life balance relative to other industries that developers could enter, the entire technology sector is rife with sexism. A survey project called "Elephant in the Valley" focused on women that held high level positions in Silicon Valley with over ten years of experience. The survey exposed that 60 % of respondents reported sexual harassment with 40% of the harassed choosing to not report out of fear that it would negatively impact their career trajectories (3). Women also had to worry about the impact of family duties affecting their career opportunities as 75% reported that family related issues and concerns were brought up in their interviews.

Despite the potential for horrendous work schedules, sexism, and other suppressed issues within the technology industry, are software developers ultimately happy with their jobs? At a macro level, 57% of all software developers reported to be reasonably happy with their jobs according to a PayScale survey, however, when looking at specialists like a network administrator the satisfaction rate dramatically rises. In the end what does all this mean for anyone that is considering becoming a software developer and for our team's project? It's clear that there is no easy way to determine if a developer will be satisfied or not with their job as there are so many factors to take into consideration. It is the team's intent to use our dataset to attempt to better understand the factors behind the reported satisfaction rates with full acknowledgement that our analysis will never fully be able to reflect reality.

1.5 References

1. "How Much Can a Computer Programmer Expect to Get Paid?" U.S. News & World Report, U.S. News & World Report.
2. Semuels, Alana. "Video Game Creators Are Burned Out and Desperate for Change." Time, Time, 11 June 2019.
3. Vassallo, Trae, et al. The Elephant in the Valley, Women in Tech.
4. Smith, Erik. "Do People Enjoy Being Software Engineers?" Surge Forward, 1 Aug. 2018.

1.6 Hypothesis

It is hypothesized that reported Work Week Hours (WWH) and reported compensation will be the strongest predictors of reported job satisfaction relative to other factors found in the survey such as country of origin, years of experience, or age. Specifically, it is hypothesized that the regression model will assign a higher weight to the compensation feature as common logic would suggest that programmers who are compensated highly are more likely to deal with long hours.

If our hypothesis holds true, we predicted that the regression model will assign a higher weight to the compensation feature as common logic would suggest that programmers who are compensated highly are more likely to deal with long hours. Cross validation methods will be utilized in conjunction with the multivariate regression in an effort to have a more concrete conclusion to our analysis of the models we develop. Due to the voluntary nature of the dataset and potential unreliability, there may be some assumptions taken by the project team although it will be clearly noted and explained where intentional manipulations of the data set were made.

1.7 Dataset(s) & Links

1. https://insights.stackoverflow.com/survey/2019

This first dataset consists of over 88,883 developers that were surveyed from 170 countries and dependent territories during 2019 by Stack Overflow. Each row includes information about the respondents in terms of their personal background, background as a developer, income, job and career satisfaction, etc. In this dataset, we will focus on featured variables such as: Country, education level, job satisfaction, type of compensation, age, number of programming languges, number of years coded,number of hours worked in a week.

2. https://insights.stackoverflow.com/survey/2020

This second dataset consists of over 64,461 developers that were surveyed from 170 countries and dependent territories during 2020 by Stack Overflow. This dataset we are going to explore has the same featured variables as explained in the previous paragraph.

1.8 Dataset Variables

The 2019 & 2020 Stack Overflow survey dataset columns we will be working with:

Column	Definition
Country	The country the respondents resides

Column	Definition
EdLevel	The Highest level of formal education
YearsCode	Including any education, number of years coding
JobSat	How respondent are satisfied with their job
CompTotal	Respondents current total compensation
CompFreq	Is compensation weekly,monthly,yearly
ConvertedComp	Salary converted to annual USD salaries (exchange rate on 2019-02-01 / 2020-02-19)
WorkWeekHrs	Respondents total working hours per week
LanguageWorkedWith	Type of programming languages respondents knows
Age	Age of respondents

1.9 Methods Used: Multivariate Regression & Cross Validation

The primary analysis technique chosen for this project was multivariate regression, and our follow-up analysis technique was cross-validation. Multivariate regression was chosen because our team was interested in comparing a software programmer's reported job satisfaction, work week hours, and compensation to identify a relationship between the variables based on our team's hypothesis.

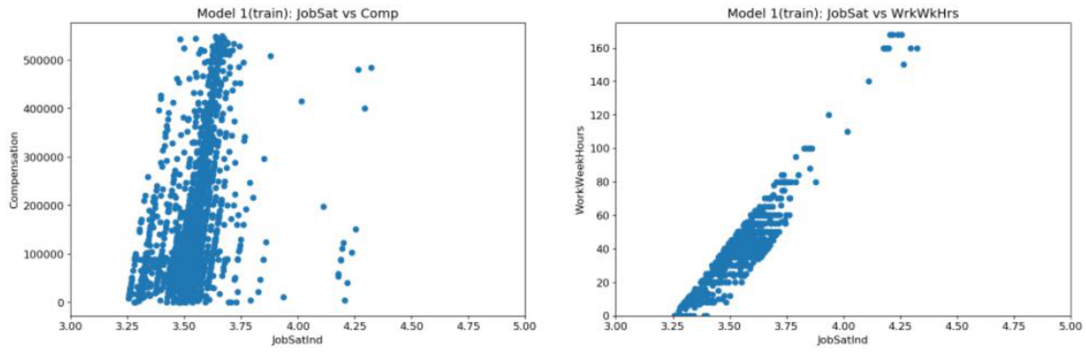
Below are the 3 Multivariate Linear Regression equations used to obtain our models 1. Work Satisfacton Index = w0 + (w1 * Compenstaion) + (w2 * WorkWeekHours) 2. Work Satisfaction Index = w0 + (w1 * Compensation) + (w2 * WorkWeekHours)^(1.2) 3. Compensation = 105,487 + (648 * WorkWeekHours) + (-486 * Number of Languages)

1.9.1 Results: Multivariate Linear Regression Model 1

1. Work Satisfacton Index = w0 + (w1 * Compenstaion) + (w2 * WorkWeekHours)

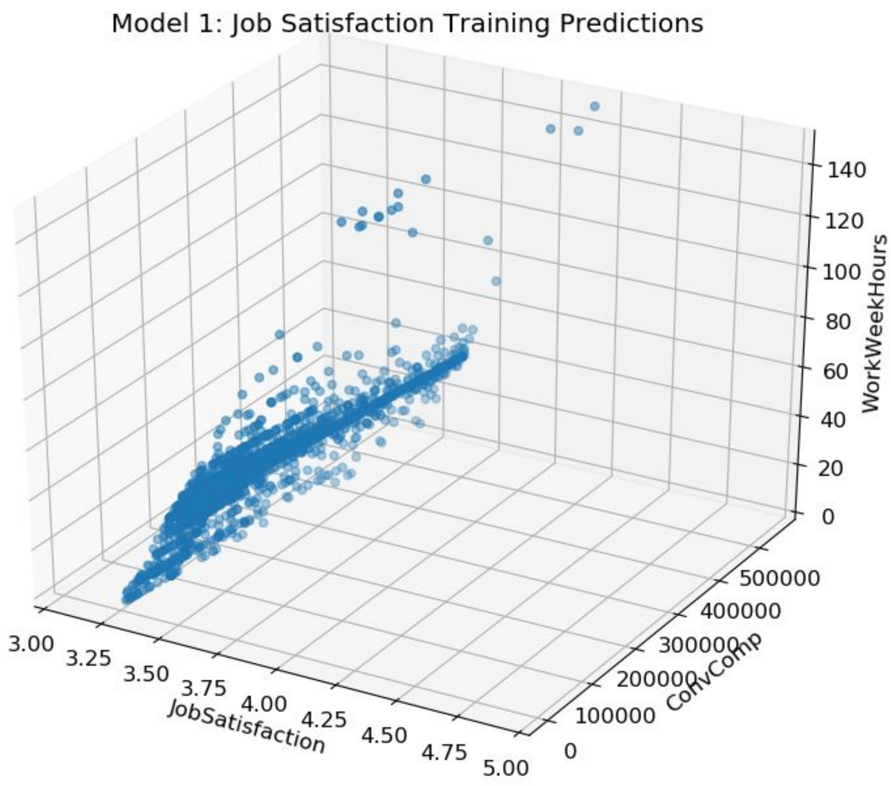
[25]:
```
Image(filename="model1.png")
```

[25]:



[26]:
```
Image(filename="model1MVR.JPG")
```

[26]:



1.9.2 Analysis: Multivariate Linear Regression Model 1

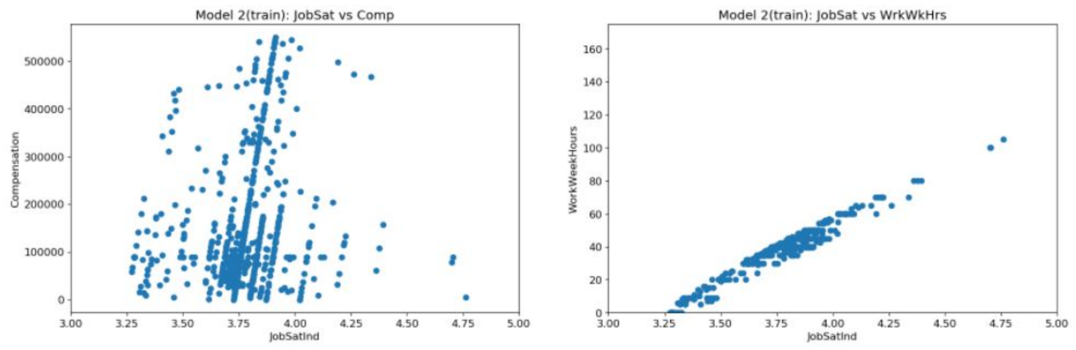
The first graph pictured above on the far top left is our model 1 prediction of the job satisfaction index against yearly compensation from the dataset.The second graph pictured above on the far top right is our model 1 prediction of the job satisfaction index against work week hours from the dataset. When both graph combined and placed into a 3-d model we then can see their interaction with each other. In these graphs you can see that the models predict job satisfaction to be at an index between 3.25 - 4.0 which is wrong. On the left picture where compensation is rated against job satisfaction, this visualization is telling that all programmers will be rating their job satisfaction to atleast 3.25 with very change despite higher and higher compensation rates which makes little sense. On the right graph, the same error is seen, the model predicts a satisfaction of around 3.50 with almost no change with regards to an increase in work week hours. Both graphs suggest that all developers will be relatively satisifed with their jobs regardless of their compensation and hours worked in a week.

1.9.3 Results: Multivariate Linear Regression Model 2

```
2. Work Satisfaction Index = w0 + (w1 * Compensation) + (w2 *
   WorkWeekHours)^(1.2)
     1. Introduced Multivariate Interaction Term as we theorize that the number
        of work week hours places more weight negativly towards the software
        developers satisfaction index.
     2. Using a 80% (training) and 20% (testing) data split for cross validation
```

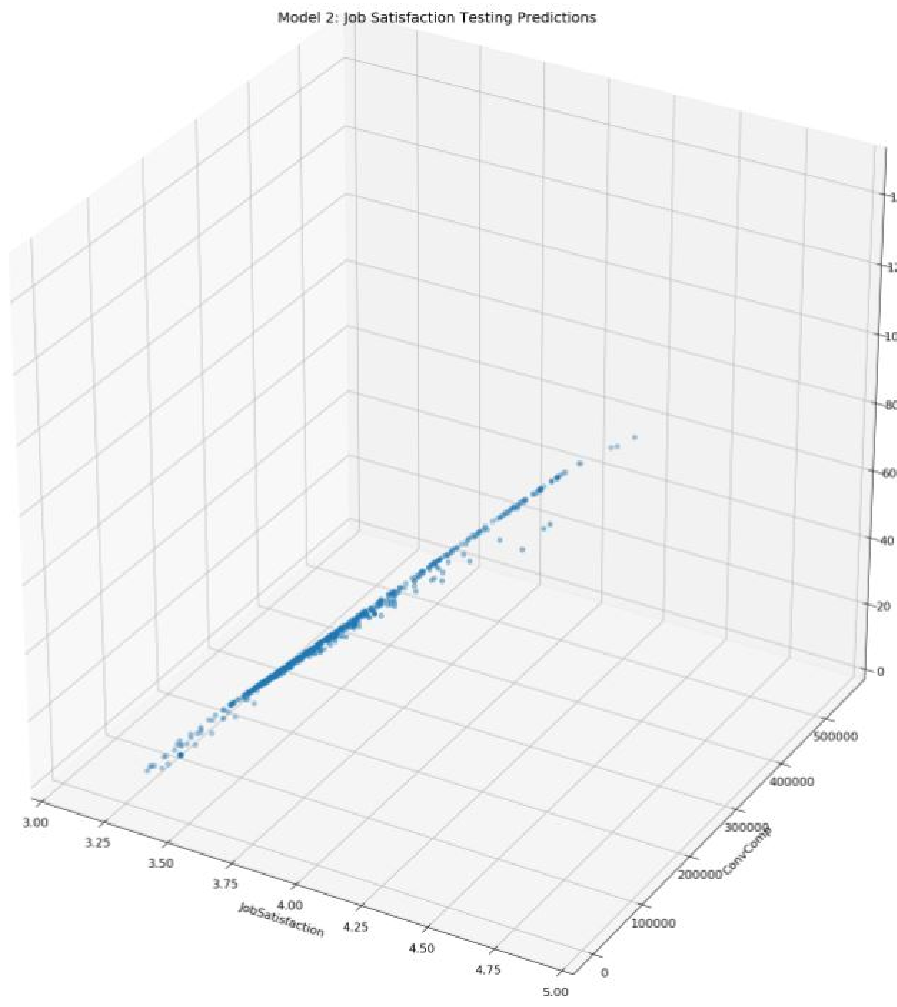
[27]:
```
Image(filename="model2.JPG")
```

[27]:



[28]:
```
Image(filename="model2MVR.JPG")
```

[28]:



1.9.4 Analysis: Multivariate Linear Regression Model 2

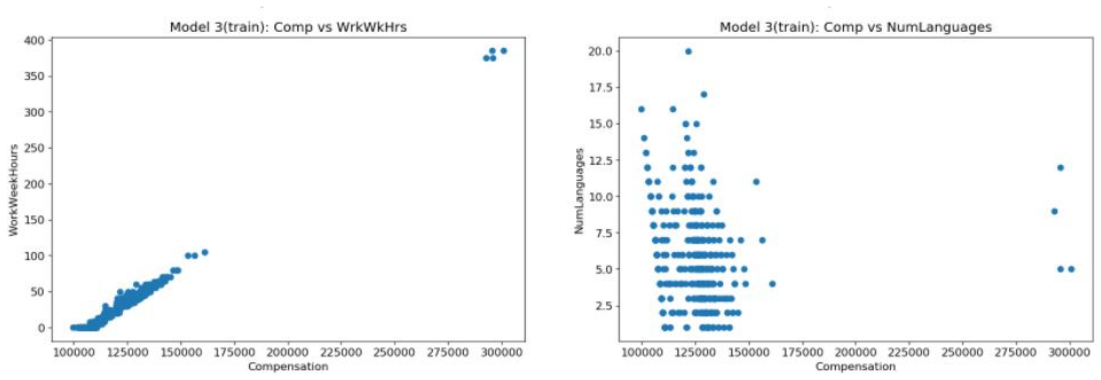
The 1st graph pictured above in the left position is our model 2 prediction of the job satisfaction index against yearly compensation from the dataset.The 2nd graph pictured above in theright position is our model 2 prediction of the job satisfaction index against work week hours from the dataset. When both graph combined and placed into a 3-d model we then can see their interaction with each other. Looking at the left graph we can see the mean job satisfaction of 3.75 and compensation varies from 0 to 500,000 dollars. The Right graph we see a potive correaltion of job satisfaction and workWeekHours, this would mean programers who works more tends to like thier job. This contractdicts our original hypothesis as devs. woulds have a higher job satisfaction with less work week hours.

1.9.5 Results: Multivariate Linear Regression Model 3

3. `Compensation = 105,487 + (648 * WorkWeekHours) + (-486 * Number of Languages)`

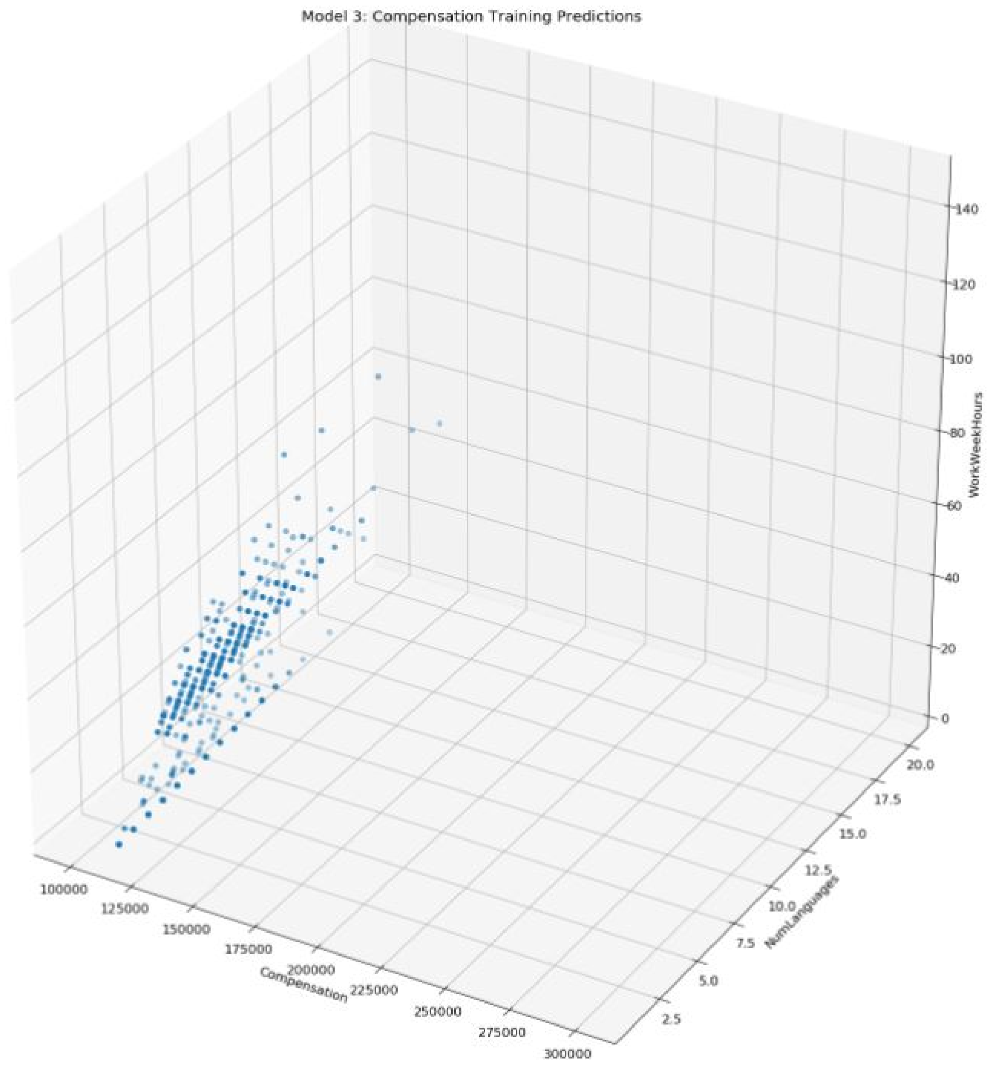
[29]:
```
Image(filename="model3.JPG")
```

[29]:



[30]:
```
Image(filename="model3MVR.JPG")
```

[30]:



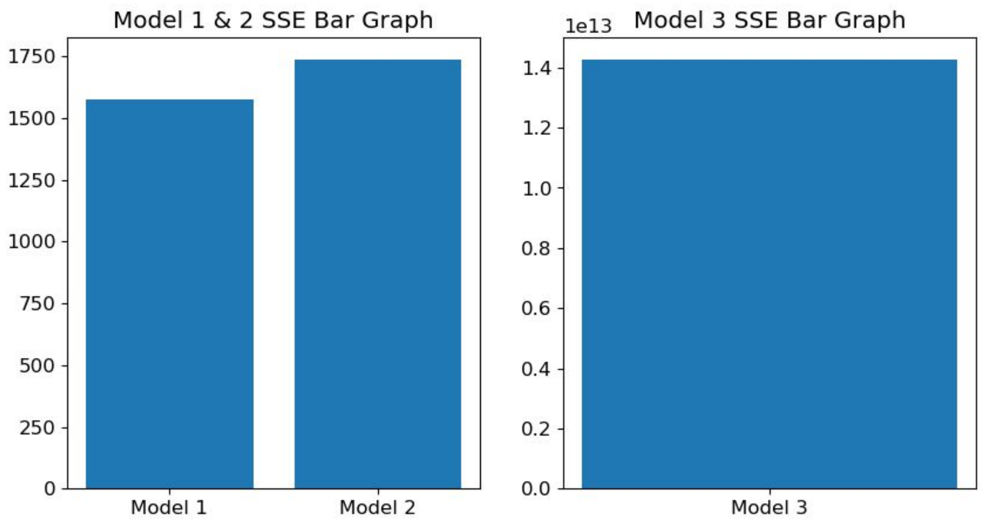
1.9.6 Analysis: Multivariate Linear Regression Model 3

The 1st graph left is our model 3 prediction of yearly compensation against work week hours from the dataset. The 2nd graph pictured above on the right is our model 3 prediction of yearly compensation against number of languages known from the dataset. When both graph combined and placed into a 3-d model we then can see their interaction with each other. Looking at left graph we can see a positve correlation to compensation and WorkWeekHours as the compensation increase so does the number of hours works. This can mean that some jobs are paying hourly and overtime is a factor as it increase the original pay rate of each programer. While other jobs, developers are being paid salary based, therfore the number of hours they each work will vary with a set salary.

1.9.7 Results: Sum Square Erros (SSE) of Models 1, 2 & 3

[31]:
```
Image(filename="SSE.JPG")
```

[31]:



1.9.8 Analysis: Sum Square Erros (SSE) of Models 1, 2 & 3

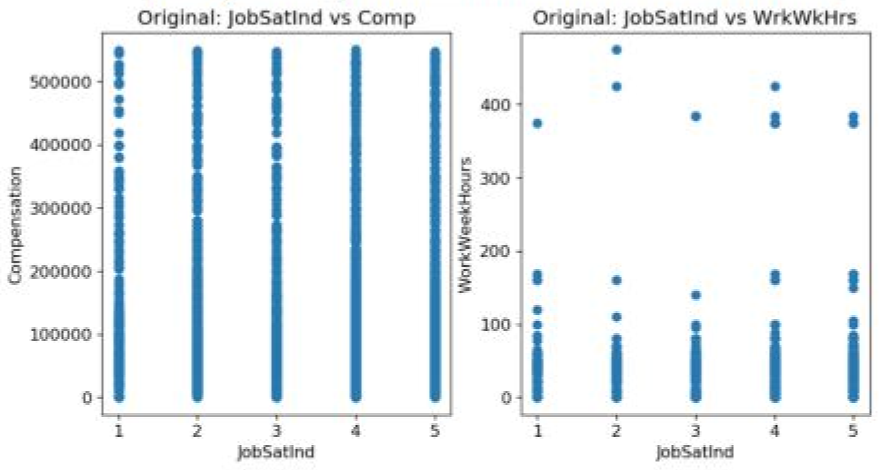
The first graph pictured above on the left is the total sum squared error from both models 1 and 2 represented in a bar format. The second graph pictured above on the right is the total sum squared error from just model 3

1.9.9 Additional Graphs

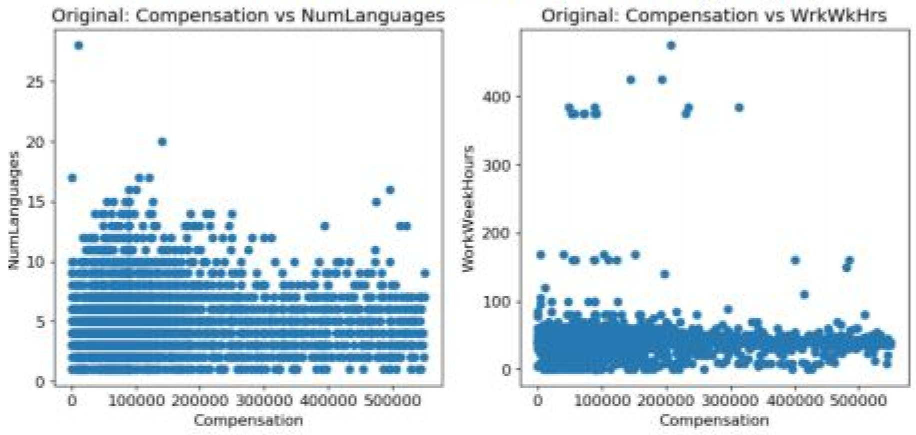
[32]:
```
Image(filename="additional.JPG")
```

[32]:

Graphing of Job Satisfaction Index in Regards to Yearly Compensation and Work Week Hours



Graphing of Compensation in Regards to Work Week Hours and Number of Computer Languages



1.10 Conclusion & Summary

At the end of this project the team has agreed to reject our original hypothesis that WorkWeekHours (WWH) and compensation are strong predictors of determining a software engineer's job satisfaction. Upon visualizing our data it appeared that there was no strong evidence that either factor performed as well as expected. When plotting job satisfaction against both compensation and WWH the number of workers seemed to be relatively normally distributed among each reported satisfaction level. This suggests that job satisfaction is a complicated and nuanced concept with a multitude of factors affecting it and was beyond our rudimentary modelling. In our exploratory background research there were hints that the question of job satisfaction would be quite complicated in terms of the variables involved so the project's results seem to align with realistic expectations.

Taking a step away from the main question of the project and touching upon our further exploration of the dataset, it appears that the answer to the question if the number of programming languages affected compensation is also a resounding no. Referencing back to the visualizations of the data, graphs show that developers who reported knowing the most languages were among some of the lowest compensated engineers. In fact, some of the highest paid programmers tended to report knowing only an average of 5 different languages which seemed to be lower than expected. A hypothesis that could be extrapolated from this analysis is that engineers who tended to specialize in a few languages and developed an expertise in the area became highly sought after and paid better.

1.11 Discussion:

Our research question was answered, however, it doesn't seem that WWH and compensation have a direct relationship with job satisfaction based on our 2 models testing job satisfaction versus WWH and compensation. We strongly believe the data analysis technique we used was a good choice given our research question, as we were deciding between regression and clustering, thus the issue instead lies with our data. There was such a varying range of survey results for each comparison that no clear trends or correlations could be seen in our graphs. For example, in the category of job satisfaction and salary, there was a mix of programmers making tons of money and disliking their jobs to programmers making tons of money and loving their jobs. And in the category of job satisfaction and long hours, there was a mix of programmers working long hours and loving their job to programmers working long hours and hating their jobs. Because of this, no clear predictions could be made via this analyzation technique. There is always the possibility that a different technique could have been as lightly better fit, but overall our team thinks this dataset had too much variability and randomness for a good, clear analysis to be made.

1.12 Privacy & Ethical Consideration

All standard privacy guidelines were followed and since the original data set did not contain any personal information there shouldn't be any privacy concerns. The team also went on to drop unnecessary information such as country of origin and gender identity as it did not need to be considered in our research going even further to anonymize the data as much as possible.

Due to the data set being from a public open source, a survey freely available on Stack Overflow, the team assumed that explicit consent was not necessary to perform analysis on the data set. Some ethical concerns that remained after trying to keep the data as anonymous as possible was the existence of several reported minors in the set. Some respondents were under the age of 18 but the majority of these outliers seemed to be intentionally facetious with responses such as being ``2 years old with a compensation of $1 billion `` and these individuals purposely removed from the data. However, several respondents appeared to be legitimate such as a 14 or 16 year old that had very little compensation and work hours and it was deemed that due to the anonymity of the data set the team would keep their responses in our analysis. Since the survey was voluntary, it was agreed that even information from these minors should be valid to use.

1.12.1 End of Report... Thank you!

1.12.2 Code Begins

1.12.3 Importing Packages

[33]:
```
#-------------------------------------------------------------#
# This Cell is dealing with importing columns packages
#-------------------------------------------------------------#

%matplotlib inline
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import os
import random
import patsy
import scipy.stats as stats
from scipy.stats import ttest_ind, chisquare, normaltest
import statsmodels.api as sm

from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings("ignore")

# set defaults
plt.style.use('default')   # seaborn custom plot style
plt.rc('figure', dpi=100, figsize=(7, 5))   # set default size/resolution
plt.rc('font', size=12)   # font size
```

1.12.4 Converting .csv files into Pandas DataFrame

[34]:
```
#------------------------------------------------------------#
# This Cell is dealing with importing .csv files
#------------------------------------------------------------#



# Make sure you rename csv file imported to match below
SO_2019_csv = 'survey_results_public2019.csv'
SO_2020_csv = 'survey_results_public2020.csv'

#covert csv into pandas data frame
SO_2019 = pd.read_csv(SO_2019_csv) #88883 Total Respondents
SO_2020 = pd.read_csv(SO_2020_csv) #64461 Total Respobdents

# Store original column names
all_var_2019 = list(SO_2019)
all_var_2020 = list(SO_2020)

# Checks for similar columns names
#print(set(all_var_2019) & set(all_var_2020))
```

1.12.5 Helper Method Functions

[35]:
```
#------------------------------------------------------------#
# This Cell is for helper functions                          #
#------------------------------------------------------------#

# Standardize YearsCode coverting to int
def standardize_years(obj):
    if ('Less' in obj):
        out = '1'
    elif ('More' in obj):
        out = '50'
    else:
        out = obj
    return out

# convert string to int to assign values
def JobSatInt(obj):
    out = obj
    if obj == 'Very dissatisfied':
        out = 1
    elif obj == 'Slightly dissatisfied':
        out = 2
    elif obj == 'Neither satisfied nor dissatisfied':
        out = 3
    elif obj == 'Slightly satisfied':
        out = 4
    elif obj == 'Very satisfied':
        out = 5
    return out

# function to count the number of programing language a respondant has
def standardize_LanguageWorkedWith(string):

        return  (string.count(';') + 1)
```

1.12.6 Data Cleaning

[36]:
```
#------------------------------------------------------------#
# This Cell is dealing with NaN values                       #
# PLEASE RUN THIS CELL ONLY ONCE, IF EDITED MUST RESTART     #
# AND RUN ALL AGAIN                                          #
#------------------------------------------------------------#



# Keep columns listed and drop the rest
SO_2019 = SO_2019[['Country', 'EdLevel', 'YearsCode', 'JobSat',
                   'CompFreq','ConvertedComp', 'WorkWeekHrs',
                   'Age', 'LanguageWorkedWith']]

SO_2020 = SO_2020[['Country', 'EdLevel', 'YearsCode', 'JobSat',
                   'CompFreq','ConvertedComp', 'WorkWeekHrs',
                   'Age','LanguageWorkedWith']]


# Merging SO_2019 and SO_2020 df
# Set ingnore_index to true to ignore index value of second df
combined_df = SO_2019.append(SO_2020, ignore_index = True)

# Drop NaN is 'Country' column (521 rows dropped)
combined_df = combined_df[combined_df['Country'].notna()]

# Replace all NaN in 'EdLevel' Column with 'Self-Taught' (Assuming Dropped out␣
 ↪of school)
combined_df['EdLevel'].fillna('Self-Taught', inplace=True)

# Replacing Nan with mean number (12) of years coded and convert YearsCode to␣
 ↪int type
dummy_df = combined_df[combined_df['YearsCode'].notna()]
```

```
dummy_df['YearsCode'] = dummy_df['YearsCode'].apply(standardize_years)
dummy_df['YearsCode'] = dummy_df['YearsCode'].astype(str).astype(int)
mean_YearCode = int(round(np.mean(dummy_df['YearsCode'])))
combined_df['YearsCode'].fillna(str(mean_YearCode), inplace=True)
combined_df['YearsCode'] = combined_df['YearsCode'].apply(standardize_years)
combined_df['YearsCode'] = combined_df['YearsCode'].astype(str).astype(int)

# Replacing Nan with mean number of JabSat and covert to int type
# (Change to setting number to neither)
combined_df['JobSat'].fillna(str('Neither satisfied nor dissatisfied'),␣
 ↪inplace=True)
combined_df['JobSatInt'] = combined_df['JobSat'].apply(JobSatInt)
combined_df['JobSatInt'] = combined_df['JobSatInt'].astype(str).astype(int)

# replace nan with 0,yearly,0 repectivley in CompTotal,CompFreq, ConvertedComp
combined_df['CompFreq'].fillna(str('Yearly'), inplace=True)
combined_df['ConvertedComp'].fillna(float(88813.35), inplace=True)
combined_df['WorkWeekHrs'].fillna(float(0.0), inplace=True)



#Drop age less than 16 (outliers and misinputteded data)
combined_df = combined_df[combined_df['Age'] >= 16]
# turning Age column from float to in
combined_df['Age'] = combined_df['Age'].astype(int)

# index reset
combined_df = combined_df.reset_index(drop=True)

yearly_ConvertedComp = []
cnt = 0
for i in combined_df.CompFreq.values:
    if i == 'Yearly':
        yearly_ConvertedComp.append(combined_df['ConvertedComp'][cnt])
        cnt += 1
    elif(i == 'Monthly'):
        yearly_ConvertedComp.append(combined_df['ConvertedComp'][cnt] * 12)
        cnt += 1
    else:
        yearly_ConvertedComp.append(combined_df['ConvertedComp'][cnt] * 52)
        cnt += 1

combined_df['yearly_ConvertedComp'] = yearly_ConvertedComp
combined_df = combined_df[combined_df['yearly_ConvertedComp'] <= 550000]

combined_df['LanguageWorkedWith'].fillna(str('None'), inplace=True)
```

```
combined_df['NumLanguage'] = combined_df['LanguageWorkedWith'].
 ↪apply(standardize_LanguageWorkedWith)

#reorder columns
combined_df = combined_df[['Country', 'EdLevel', 'YearsCode', 'JobSat',␣
 ↪'JobSatInt',
        'CompFreq', 'ConvertedComp', 'yearly_ConvertedComp', 'WorkWeekHrs',
       'Age', 'LanguageWorkedWith', 'NumLanguage']]
```

2 Multivariate Linear Regression Model 1:

`Work Satisfaction Index = w0 + w1 x Compensation + w2 x WorkWeekHours`

3 Multivariate Linear Regression Model 2:

`Work Satisfaction Index = w0 + w1 x Compensation + w2 x WorkWeekHours^(1.2)`

- How does compensation and WorkWeekHours predict Job Satisfaction levels? Might want to use Multivariate Interaction terms?
- Using a 80% (training) and 20% (testing) data split for Cross Validation

4 Multivariate Linear Regression Model 3:

`Compensation = 105,487 + 648 * WorkWeekHrs + (-486) * NumLanguages`

4.0.1 Random Sampling

[37]:
```
#------------------------------------------------------------#
# This cell is taking a subset of the original data          #
#       5000 instances for sample training                   #
#       1000 instances  for sample testing                   #
#------------------------------------------------------------#

# ranomly select 6000 elements from dataset and rerun 1000 times
# to make it more random
for i in range(100):
    total_random_sample = combined_df.sample(n=6000)

total_random_sample = total_random_sample.reset_index(drop=True)

# 6000 elem randomly selected to test
Random_df = total_random_sample.head(6000)
```

4.0.2 Data Manipulation & Regression Fitting/Training

[38]:
```
#------------------------------------------------------------#
# This cell is dealing with the Multivariate Regression      #
# Data Manipulation and Regression fitting/training          #
#------------------------------------------------------------#

# Extracting variables from the Data into Numpy Array Values
Y = Random_df['JobSatInt'].values
X1 = Random_df['yearly_ConvertedComp'].values
X2 = Random_df['WorkWeekHrs'].values
X3 = Random_df['NumLanguage'].values

X = Random_df[['yearly_ConvertedComp', 'WorkWeekHrs', 'NumLanguage']]
Y = Random_df['JobSatInt']
X2 = Random_df[['WorkWeekHrs', 'NumLanguage']]
Y2 = Random_df['yearly_ConvertedComp']

# Cross-validation
# split the numpy array data into 80% training split and 20% testing split
split80 = int(len(X) * .8)
split20 = int(len(X) * .2)
Xtrain = X.values[0:split80,]
Xtrain2 = X2.values[0:split80,]

Xtest = X.values[split80:,]
Xtest2 = X2.values[split80:,]

Ytrain = Y.values[0:split80,]
Ytrain2 = Y2.values[0:split80,]

Ytest = Y.values[split80:,]
Ytest2 = Y2.values[split80:,]

# numpy ones array for the regression training
# Put data into matrix Array form and solve for the weight vector
ones = np.ones(split80)
Xtrain1 = np.column_stack((ones, Xtrain))
w = np.linalg.lstsq(Xtrain1, Ytrain, rcond = None)[0]

Xtrain3 = np.column_stack((ones, Xtrain2))
w2 = np.linalg.lstsq(Xtrain3, Ytrain2, rcond = None)[0]

# This first model is just a basic Linear Regression model with the weights␣
↪multiplying
# their respective dimensions/features within the data
```

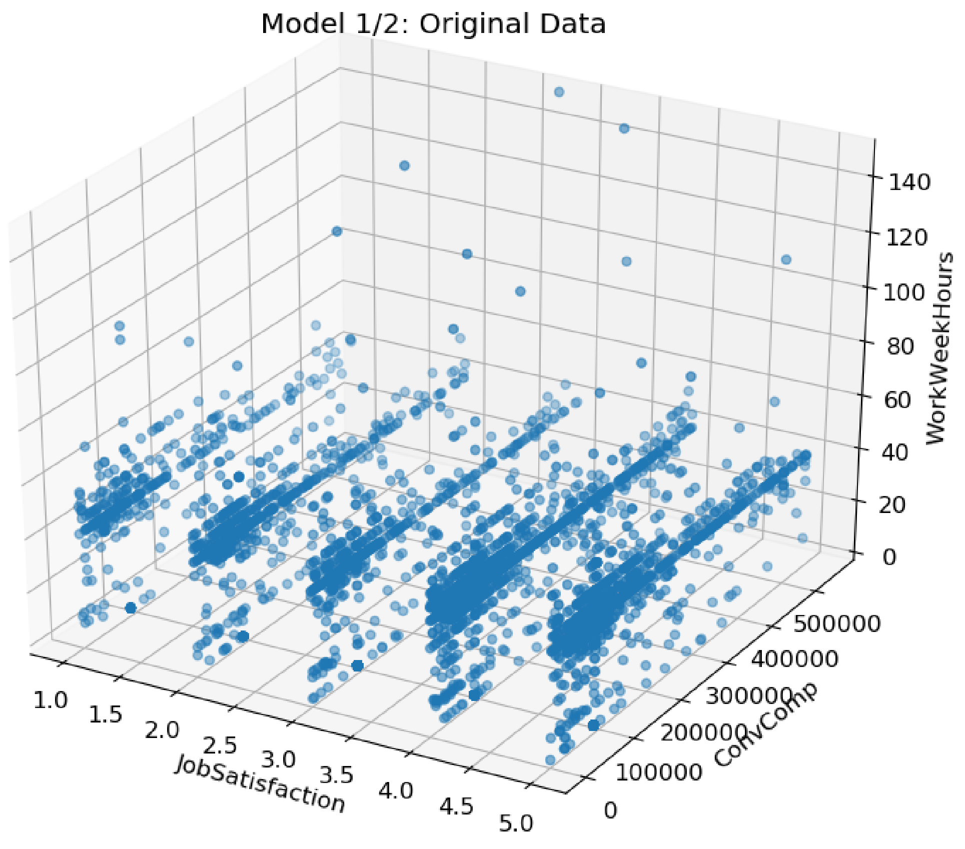
```
print('Regression Model 1: JobSat = 3.2 + 2.68e-9 * Compensation + .008 *␣
 ↪WorkWeekHrs')
JobSat_train = w[0] + w[1] * Xtrain[:,0] + w[2] * Xtrain[:,1]
JobSat_test = w[0] + w[1] * Xtest[:,0] + w[2] * Xtest[:,1]

# This second model is taking more weight on the Work Week Hours dimension of␣
 ↪the data.
# You can see this by observing that the Work Week Hours column is to the 1.2th␣
 ↪power.
print('Regression Model 2: JobSat = 3.2 + 2.68e-9 * Compensation + .008 *␣
 ↪WorkWeekHrs * Compensation')
JobSat_train2 = w[0] + w[1] * Xtrain[:,0] + w[2] * Xtrain[:,1]**(1.2)
JobSat_test2 = w[0] + w[1] * Xtest[:,0] + w[2] * Xtest[:,1]**(1.2)

# This Third model is another basic Linear Regression model but is dealing with␣
 ↪more
# data exploration. In this model we are exploring if we can predict␣
 ↪Compensation via
# Work Week Hours and Number Of Languages known.
print('Regression Model 3: Compensation = 105,487 + 648 * WorkWeekHrs + (-486)␣
 ↪* NumLanguages')
JobSat_train3 = w2[0] + w2[1] * Xtrain2[:,0] + w2[2] * Xtrain2[:,1]
JobSat_test3 = w2[0] + w2[1] * Xtest2[:,0] + w2[2] * Xtest2[:,1]
```

```
Regression Model 1: JobSat = 3.2 + 2.68e-9 * Compensation + .008 * WorkWeekHrs
Regression Model 2: JobSat = 3.2 + 2.68e-9 * Compensation + .008 * WorkWeekHrs *
Compensation
Regression Model 3: Compensation = 105,487 + 648 * WorkWeekHrs + (-486) *
NumLanguages
```

4.0.3 Plotting of Original Data (Model 1 & 2)

[39]:
```
#-----------------------------------------------------------#
# This cell is dealing with the plotting of the original and #
# unaltered data in a 3D plot for all 3 dimensions to be     #
# shown of model 1 and 2                                     #
#-----------------------------------------------------------#



# Plotting the original data
# 3D Plot of the original data points against eachother
plt.rcParams['figure.figsize'] = [10, 8]
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(Y, X['yearly_ConvertedComp'], X['WorkWeekHrs']);
ax.set_xlabel('JobSatisfaction')
ax.set_ylabel('ConvComp')
ax.set_zlabel('WorkWeekHours');
ax.set_title('Model 1/2: Original Data')
ax.set_zlim(0, 150)

plt.show
```

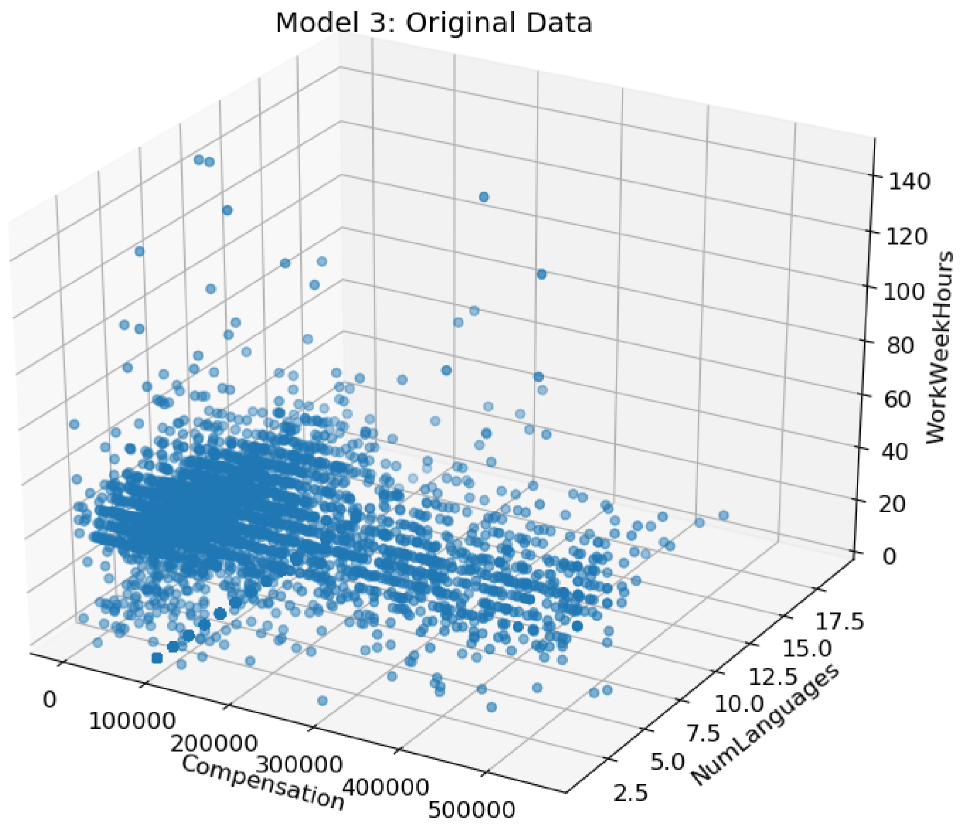
[39]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



4.0.4 Plotting Original Data (Model 3)

[40]:
```
#-----------------------------------------------------------#
# This cell is dealing with the plotting of the original and #
# unaltered data in a 3D plot for all 3 dimensions to be     #
# shown of model 3                                           #
#-----------------------------------------------------------#


# Plotting the original data
# 3D Plot of the original data points against eachother
plt.rcParams['figure.figsize'] = [10, 8]
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(X['yearly_ConvertedComp'], X['NumLanguage'], X['WorkWeekHrs']);
ax.set_xlabel('Compensation')
ax.set_zlabel('WorkWeekHours')
ax.set_ylabel('NumLanguages');
ax.set_title('Model 3: Original Data')
ax.set_zlim(0,150)

plt.show
```

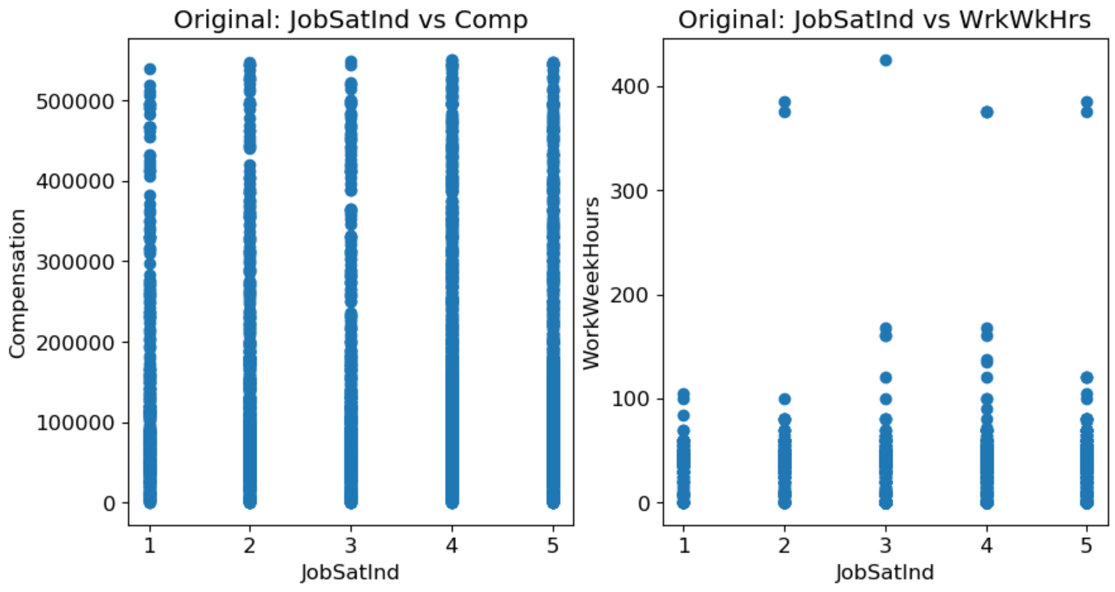
```
[40]: <function matplotlib.pyplot.show(*args, **kw)>
```



4.0.5 Graphing of Job Satisfaction Index in Regards to Yearly Compensation and Work Week Hours

```
[41]: #------------------------------------------------------------#
      # This cell is dealing with the graphing of the Job          #
      # Satisfaction Index in regards to Yearly Compensation and   #
      # Work Week Hours. All the data being graphed in this cell   #
      # is original data, not altered in any way                  #
      #------------------------------------------------------------#

      # Plotting the original data along with the model generated data
      plt.rcParams['figure.figsize'] = [10, 5]

      fig, ax = plt.subplots(1,2)

      # Plotting real Job Satisfaction data against Compensation data
      ax[0].scatter(Y, X['yearly_ConvertedComp'])
      ax[0].set_xlabel('JobSatInd')
      ax[0].set_ylabel('Compensation')
      ax[0].set_title('Original: JobSatInd vs Comp')

      # Plotting real Job Satisfaction data against Work Week Hours data
      ax[1].scatter(Y, X['WorkWeekHrs'])
      ax[1].set_xlabel('JobSatInd')
      ax[1].set_ylabel('WorkWeekHours')
      ax[1].set_title('Original: JobSatInd vs WrkWkHrs')
```

```
[41]: Text(0.5, 1.0, 'Original: JobSatInd vs WrkWkHrs')
```



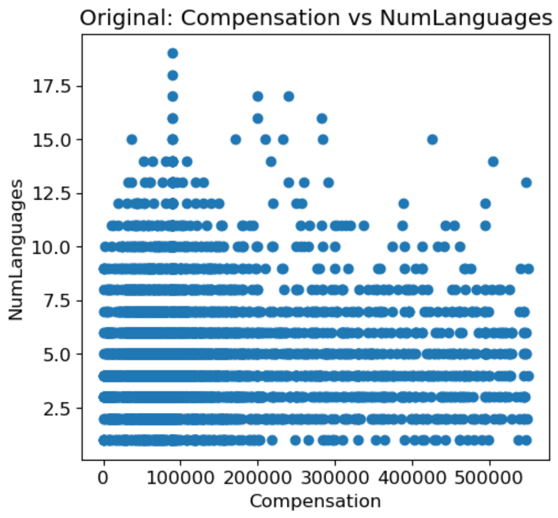
4.0.6 Graph Explanations

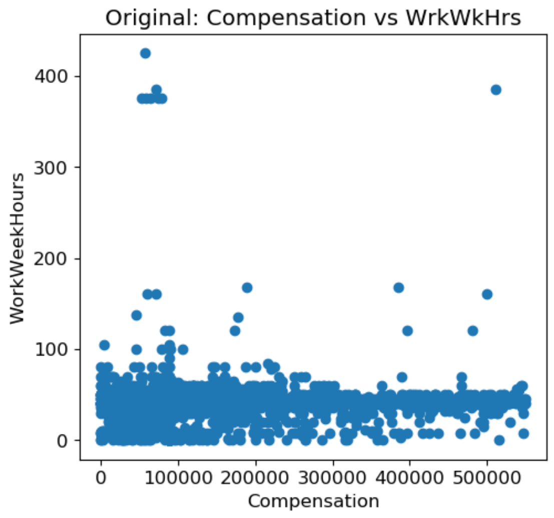
- The first graph pictured above on the left is the original data representation of the job satisfaction index against yearly compensation from the dataset
- The second graph pictured above on the right is the original data representation of the job satisfaction index against work week hours from the dataset

4.0.7 Graphing of Compensation in Regards to Work Week Hours and Number of Computer Languages

[42]:
```
#------------------------------------------------------------#
# This cell is dealing with the graphing of Compensation     #
# in regards to Work Week Hours and Num Languages All the    #
# data being graphed in this cell is original data, not      #
# altered in any way                                         #
#------------------------------------------------------------#

# Plotting the original data along with the model generated data
plt.rcParams['figure.figsize'] = [12, 5]

fig, ax = plt.subplots(1,2)

# Plotting real Job Satisfaction data against Compensation data
ax[0].scatter(X['yearly_ConvertedComp'], X['NumLanguage'])
ax[0].set_xlabel('Compensation')
ax[0].set_ylabel('NumLanguages')
ax[0].set_title('Original: Compensation vs NumLanguages')

# Plotting real Job Satisfaction data against Work Week Hours data
ax[1].scatter(X['yearly_ConvertedComp'], X['WorkWeekHrs'])
ax[1].set_xlabel('Compensation')
ax[1].set_ylabel('WorkWeekHours')
ax[1].set_title('Original: Compensation vs WrkWkHrs')
```

[42]:
```
Text(0.5, 1.0, 'Original: Compensation vs WrkWkHrs')
```





4.0.8 Graph Explanations

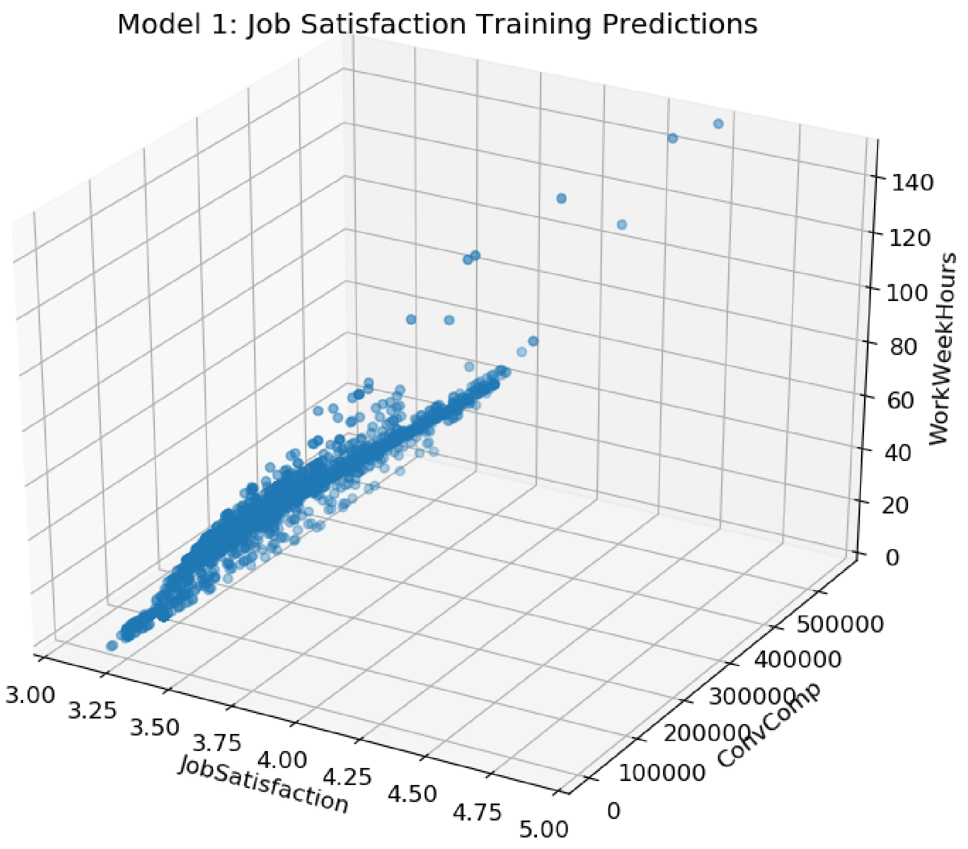
- The first graph pictured above on the left is the original data representation of yearly compensation against the number of languages known from the dataset
- The second graph pictured above on the right is the original data representation of yearly compensation against work week hours from the dataset

4.0.9 Plotting of Model 1 Training Predictions

[43]:
```
#-------------------------------------------------------------#
# This cell is dealing with the plotting of model 1 training #
# predictions for Job Satisfaction data against Work Week     #
# Hours and Compensation in a 3D plot for all 3 dimensions    #
# to be shown                                                 #
#-------------------------------------------------------------#

plt.rcParams['figure.figsize'] = [10, 8]
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(JobSat_train, Xtrain[:,0], Xtrain[:,1], cmap='Greens');
ax.set_xlabel('JobSatisfaction')
ax.set_ylabel('ConvComp')
ax.set_zlabel('WorkWeekHours');
ax.set_title('Model 1: Job Satisfaction Training Predictions')
ax.set_zlim(0, 150)
ax.set_xlim(3, 5)
```

```
plt.show
```

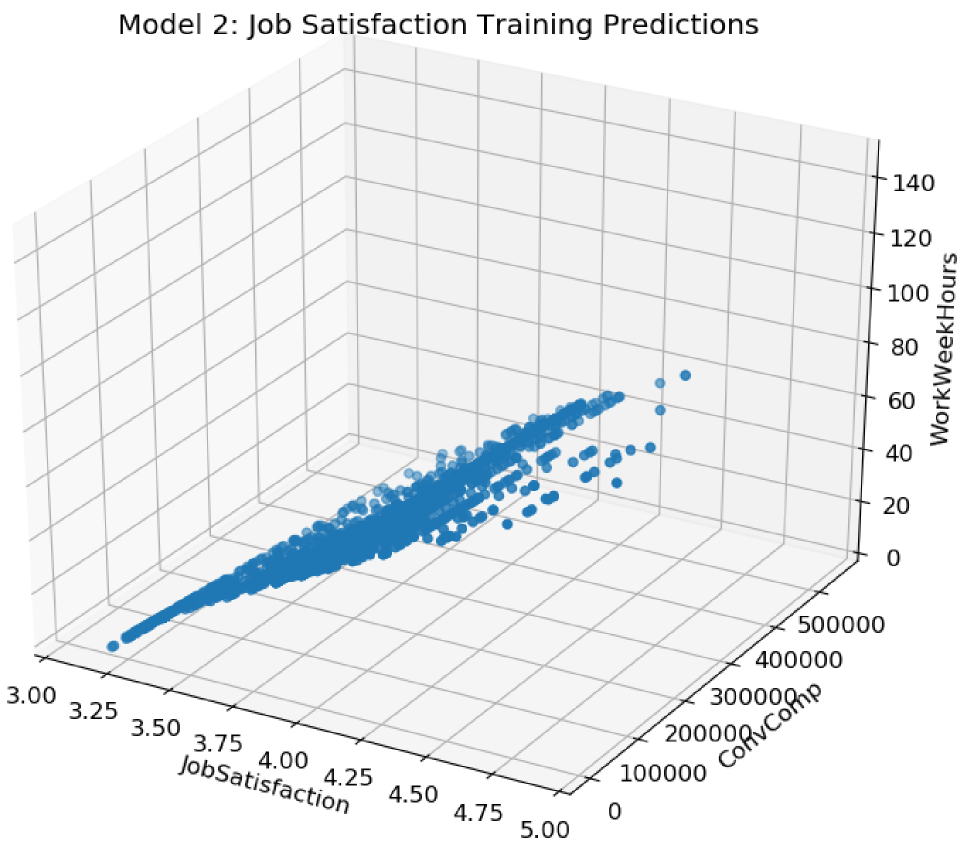
[43]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



4.0.10 Plotting of Model 2 Training Prediction for Job Satisfaction Data against Work Week Hours and Compensation

[44]:
```
#------------------------------------------------------------#
# This cell is dealing with the plotting of model 2 training #
# predictions for Job Satisfaction data against Work Week    #
# Hours and Compensation in a 3D plot for all 3 dimensions   #
# to be shown                                                #
#------------------------------------------------------------#
counter = 0
for i in JobSat_train2:
    if i > 5:
        JobSat_train2[counter] = 0
    counter += 1


plt.rcParams['figure.figsize'] = [10, 8]
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(JobSat_train2, Xtrain[:,0], Xtrain[:,1], cmap='Greens');
ax.set_xlabel('JobSatisfaction')
ax.set_ylabel('ConvComp')
ax.set_zlabel('WorkWeekHours');
ax.set_title('Model 2: Job Satisfaction Training Predictions')
ax.set_zlim(0, 150)
ax.set_xlim(3, 5)

plt.show
```

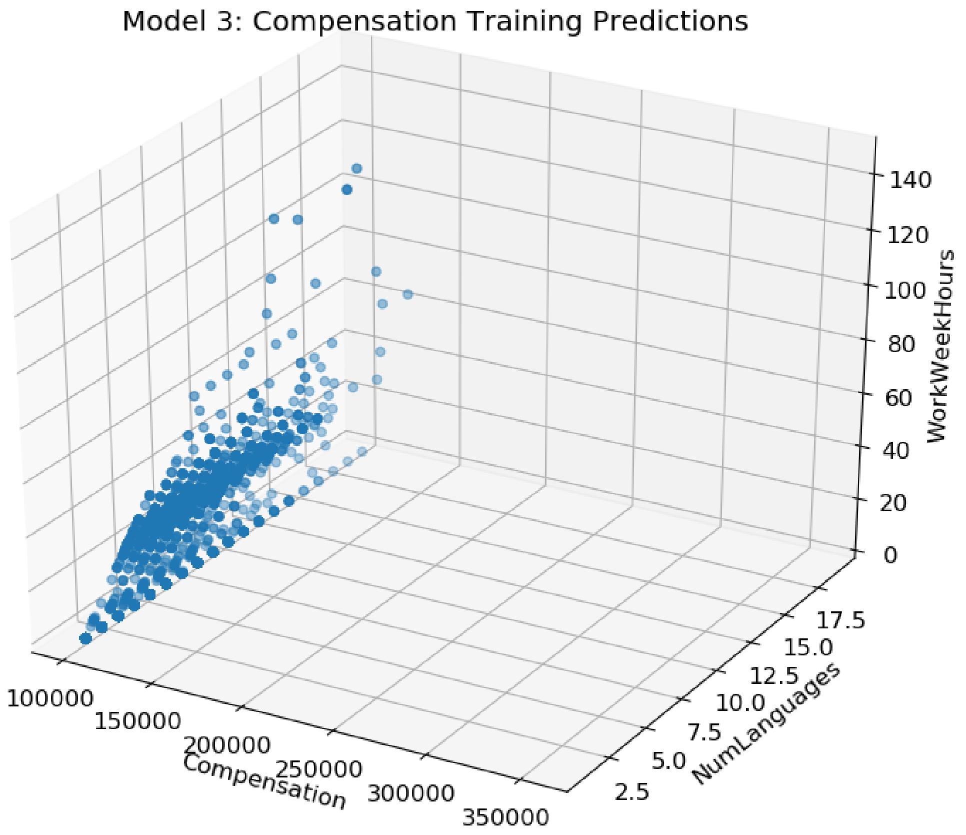
```
[44]: <function matplotlib.pyplot.show(*args, **kw)>
```



4.0.11 Plotting of Model 3 Training Predictions for Compensation Data Against Work Week Hours and Number of Computer Languages

[45]:
```
#------------------------------------------------------------#
# This cell is dealing with the plotting of model 3 training #
# predictions for Compensation data against Work Week        #
# Hours and NumLanguages in a 3D plot for all 3 dimensions   #
# to be shown                                                #
#------------------------------------------------------------#
# Xtrain2[:,0] is work week hours
# Xtrain2[:,1] is Num Languages

plt.rcParams['figure.figsize'] = [10, 8]
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(JobSat_train3, Xtrain2[:,1], Xtrain2[:,0]);
ax.set_xlabel('Compensation')
ax.set_ylabel('NumLanguages')
ax.set_zlabel('WorkWeekHours');
ax.set_title('Model 3: Compensation Training Predictions')
ax.set_zlim(0,150);

plt.show
```

[45]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```

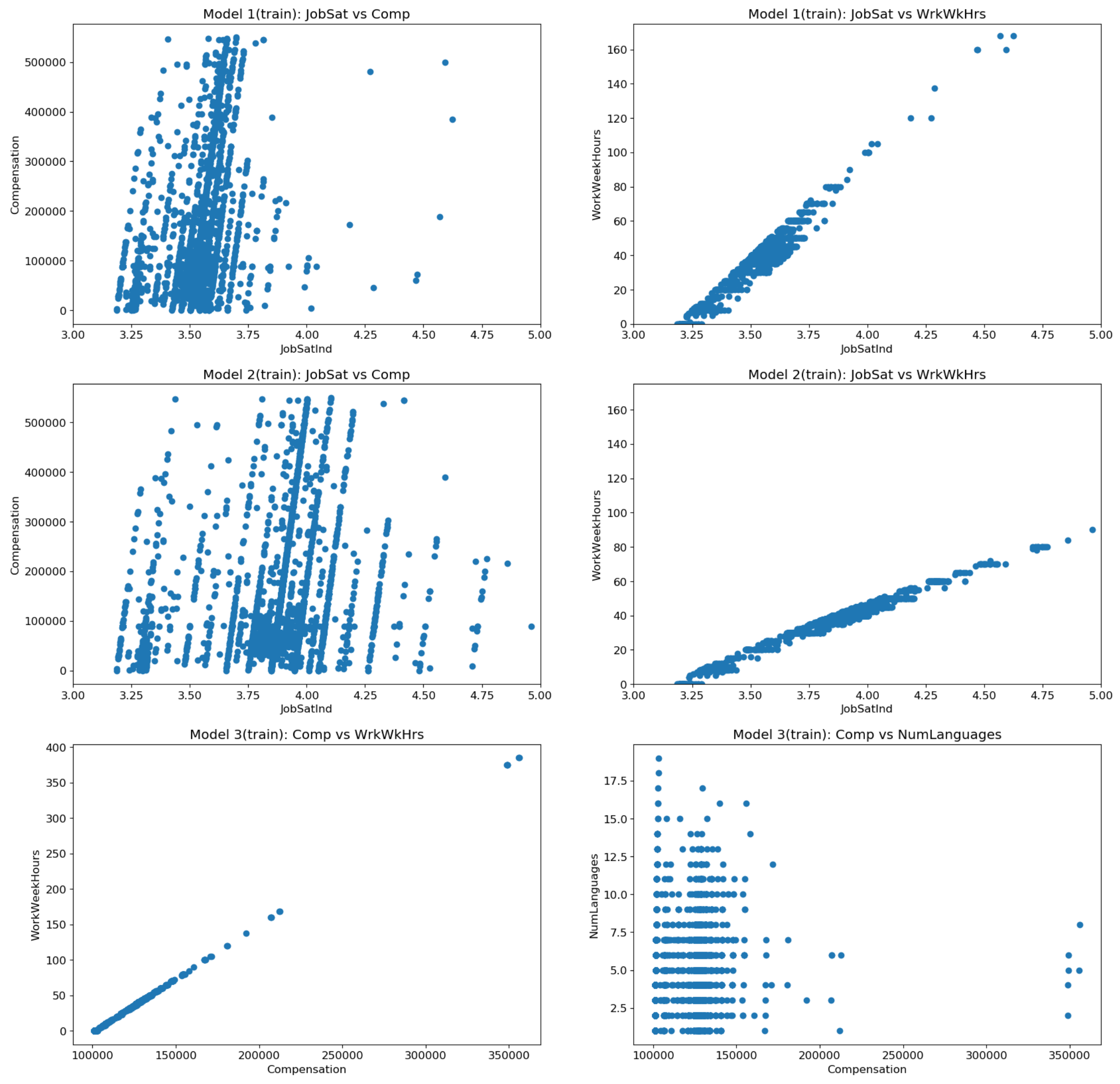


4.0.12 Graphing of Training Data of All Models and their Respective Data

[46]:
```
#-------------------------------------------------------------#
# This cell is dealing with the graphing of the training data #
# of all models and their respective data. All the data being #
# graphed in this cell is being plotted against the respective#
# model predictions                                           #
#-------------------------------------------------------------#



# Plotting the original data along with the model generated data
plt.rcParams['figure.figsize'] = [20, 20]

fig, ax = plt.subplots(3,2)

X = Random_df[['yearly_ConvertedComp','WorkWeekHrs']]
Y = Random_df['JobSatInt']
```

```python
# Plotting Model 1 training Job Satisfaction predictions against Compensation
ax[0,0].scatter(JobSat_train, Xtrain[:,0])
ax[0,0].set_xlabel('JobSatInd')
ax[0,0].set_ylabel('Compensation')
ax[0,0].set_xlim(3,5)
ax[0,0].set_title('Model 1(train): JobSat vs Comp')

# Plotting Model 1 training Job Satisfaction predictions against Work Week Hours
ax[0,1].scatter(JobSat_train, Xtrain[:,1])
ax[0,1].set_xlabel('JobSatInd')
ax[0,1].set_ylabel('WorkWeekHours')
ax[0,1].set_xlim(3,5)
ax[0,1].set_ylim(0,175)
ax[0,1].set_title('Model 1(train): JobSat vs WrkWkHrs')

# Plotting Model 2 training Job Satisfaction predictions against Compensation
ax[1,0].scatter(JobSat_train2, Xtrain[:,0])
ax[1,0].set_xlabel('JobSatInd')
ax[1,0].set_ylabel('Compensation')
ax[1,0].set_xlim(3,5)
ax[1,0].set_title('Model 2(train): JobSat vs Comp')

# Plotting Model 2 training Job Satisfaction predictions against Work Week Hours
ax[1,1].scatter(JobSat_train2, Xtrain[:,1])
ax[1,1].set_xlabel('JobSatInd')
ax[1,1].set_ylabel('WorkWeekHours')
ax[1,1].set_xlim(3,5)
ax[1,1].set_ylim(0,175)
ax[1,1].set_title('Model 2(train): JobSat vs WrkWkHrs')

# Plotting Model 3 training Comensation predictions against Work Week Hours
ax[2,0].scatter(JobSat_train3, Xtrain2[:,0])
ax[2,0].set_xlabel('Compensation')
ax[2,0].set_ylabel('WorkWeekHours')
ax[2,0].set_title('Model 3(train): Comp vs WrkWkHrs')

# Plotting Model 3 training Compensation predictions against NumLanguages
ax[2,1].scatter(JobSat_train3, Xtrain2[:,1])
ax[2,1].set_xlabel('Compensation')
ax[2,1].set_ylabel('NumLanguages')
ax[2,1].set_title('Model 3(train): Comp vs NumLanguages')
```

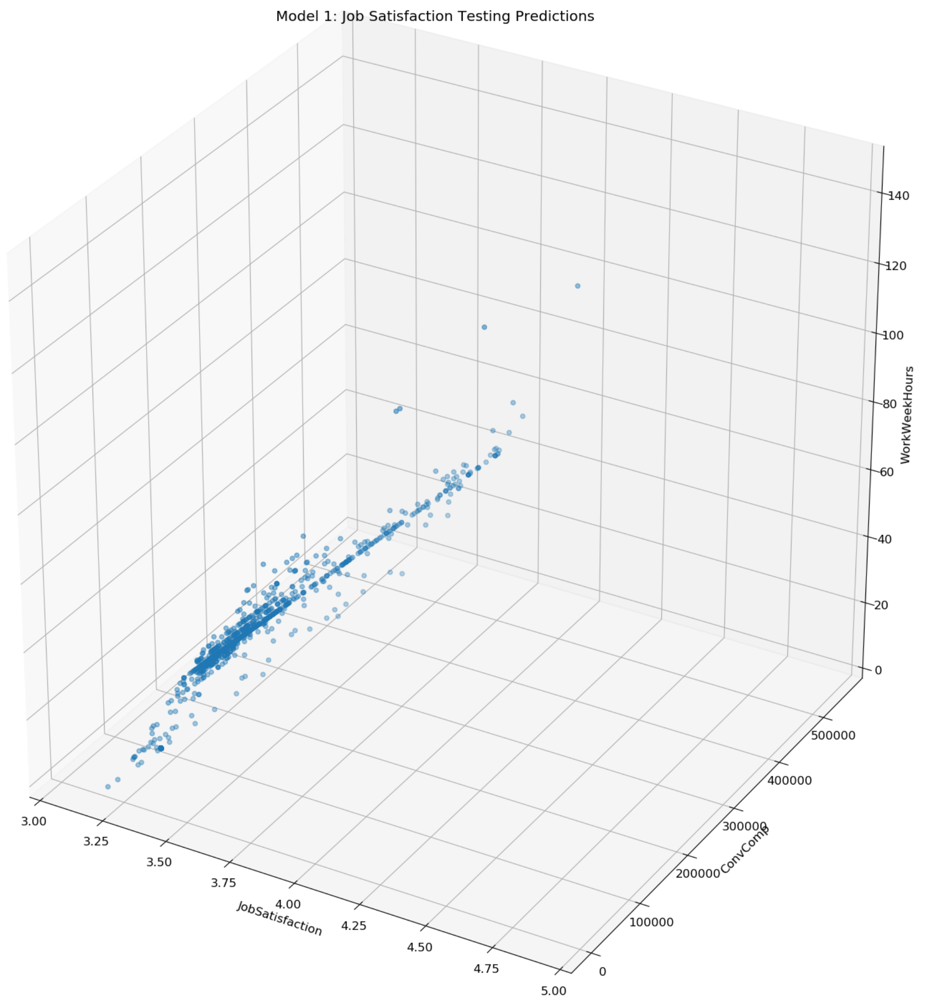
```
[46]: Text(0.5, 1.0, 'Model 3(train): Comp vs NumLanguages')
```



4.0.13 Plotting of Model 1 Testing Prediction for Job Satisfaction Data Against Work Week Hours and Compensation

[47]:
```
#-------------------------------------------------------------#
# This cell is dealing with the plotting of model 1 testing   #
# predictions for Job Satisfaction data against Work Week     #
# Hours and Compensation in a 3D plot for all 3 dimensions    #
# to be shown                                                 #
#-------------------------------------------------------------#

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(JobSat_test, Xtest[:,0], Xtest[:,1], cmap='Greens');
ax.set_xlabel('JobSatisfaction')
ax.set_ylabel('ConvComp')
ax.set_zlabel('WorkWeekHours');
ax.set_title('Model 1: Job Satisfaction Testing Predictions ');
ax.set_zlim(0, 150)
ax.set_xlim(3, 5)

plt.show
```

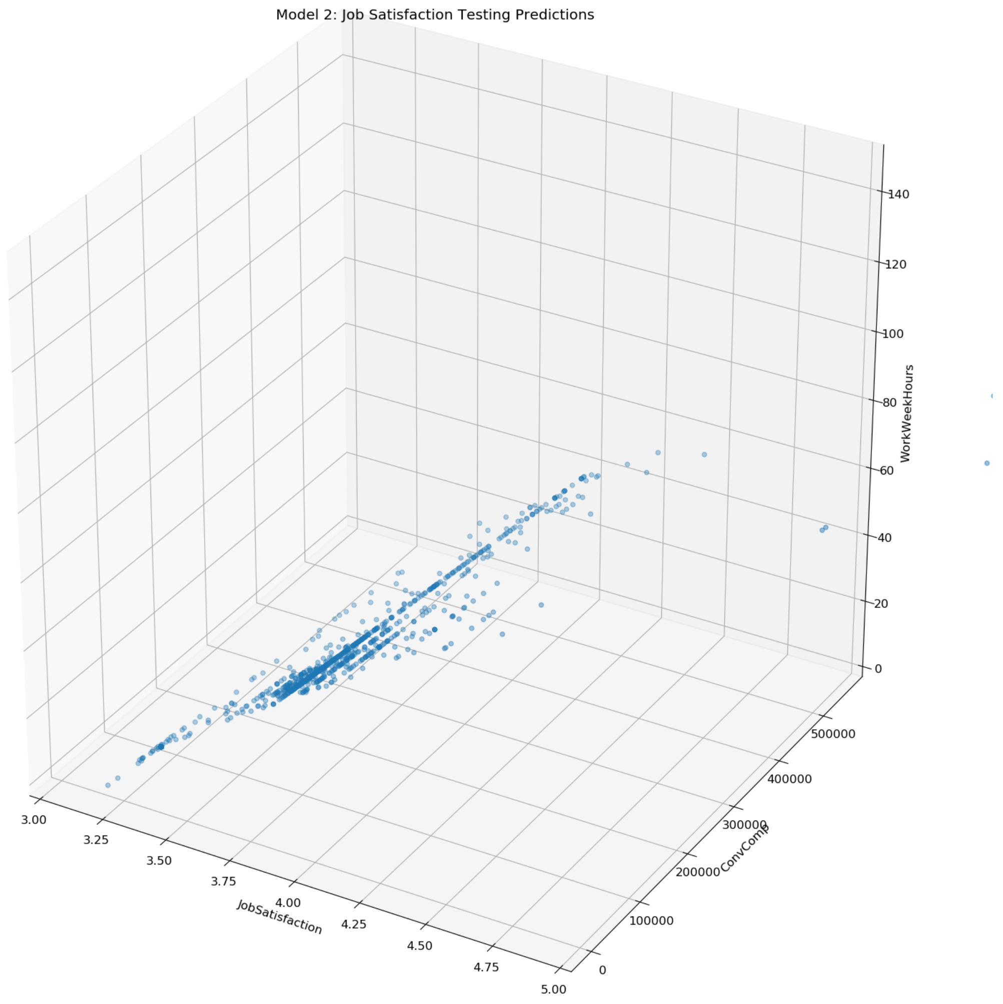
```
[47]: <function matplotlib.pyplot.show(*args, **kw)>
```



4.0.14 PLotting of Model 2 Testing Predictions for Job Satisfaction Data Against Work Week Hours and Compensation

[48]:
```
#-------------------------------------------------------------#
# This cell is dealing with the plotting of model 2 testing   #
# predictions for Job Satisfaction data against Work Week     #
# Hours and Compensation in a 3D plot for all 3 dimensions    #
# to be shown                                                 #
#-------------------------------------------------------------#

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(JobSat_test2, Xtest[:,0], Xtest[:,1], cmap='Greens');
ax.set_xlabel('JobSatisfaction')
ax.set_ylabel('ConvComp')
ax.set_zlabel('WorkWeekHours');
ax.set_title('Model 2: Job Satisfaction Testing Predictions ');
ax.set_zlim(0, 150)
ax.set_xlim(3, 5)

plt.show
```

[48]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```

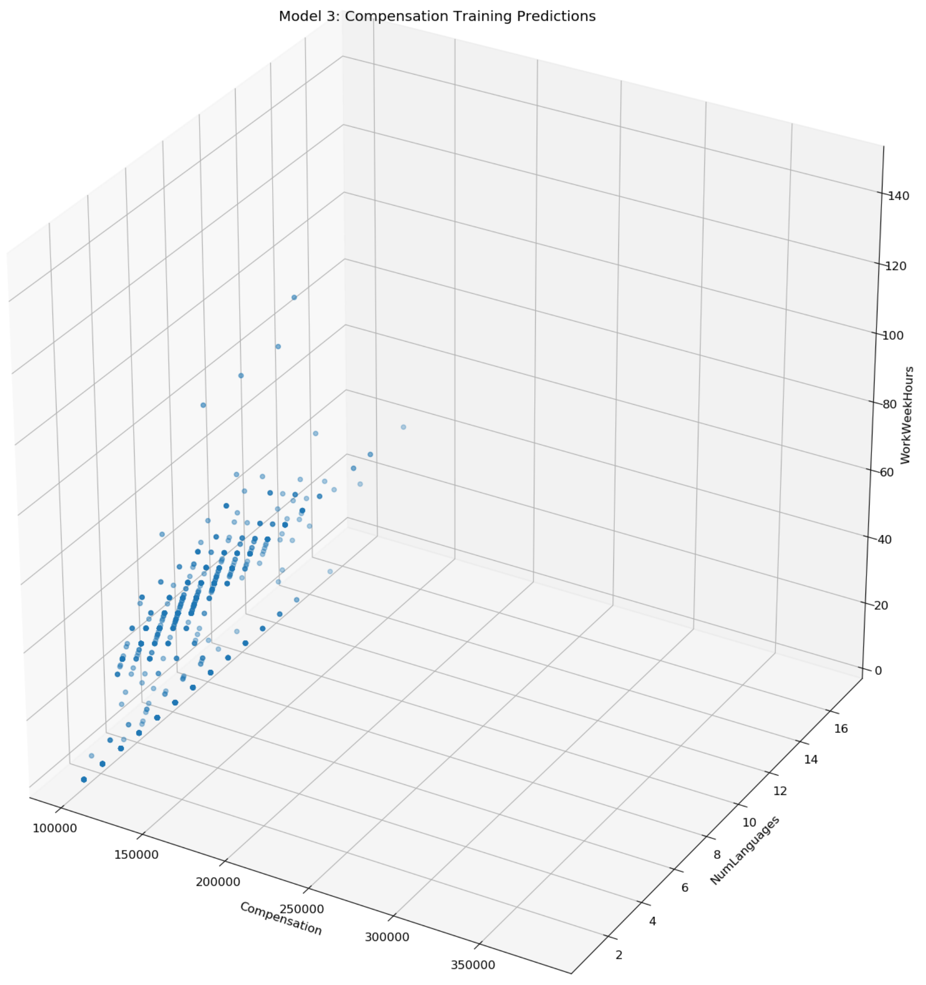


4.0.15 Plotting of Model 3 Testing Predictions for Compensation Data Against Work Week Hours and Number of Languages

[49]:
```
#-------------------------------------------------------------#
# This cell is dealing with the plotting of model 3 testing  #
# predictions for Compensation data against Work Week Hours  #
# and NumLanguages in a 3D plot for all 3 dimensions   #
# to be shown                                                 #
#-------------------------------------------------------------#

fig = plt.figure()
ax = plt.axes(projection='3d')
```

```
ax.scatter3D(JobSat_test3, Xtest2[:,1], Xtest2[:,0]);
ax.set_xlabel('Compensation')
ax.set_ylabel('NumLanguages')
ax.set_zlabel('WorkWeekHours');
ax.set_title('Model 3: Compensation Training Predictions')
ax.set_zlim(0,150);

plt.show
```

```
[49]: <function matplotlib.pyplot.show(*args, **kw)>
```

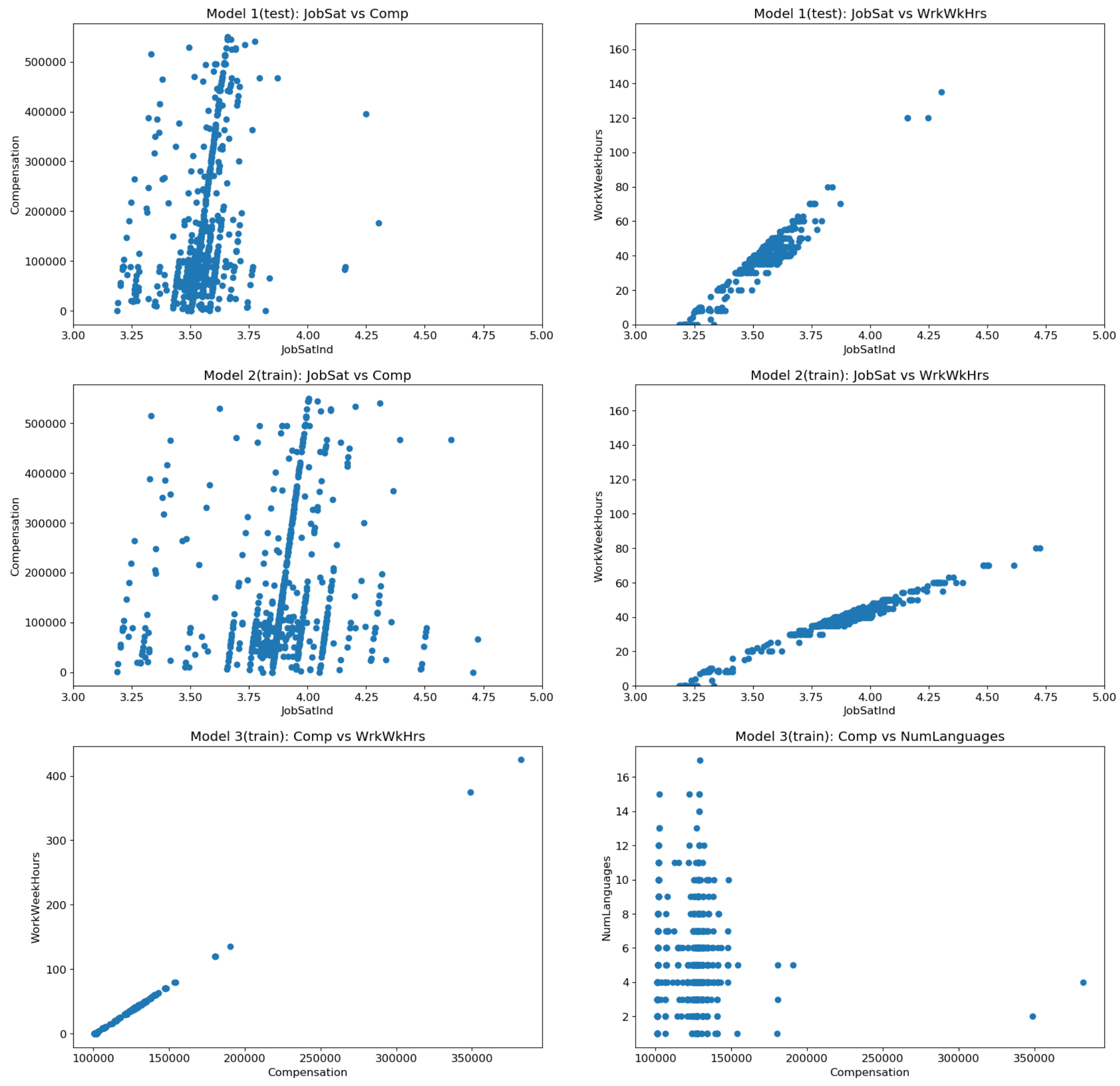


4.0.16 Graphing of Testing Data of All Models and their Respective Data

[50]:
```
#--------------------------------------------------------------#
# This cell is dealing with the graphing of the testing data  #
# of all models and their respective data. All the data being #
# graphed in this cell is being plotted against the respective#
# model predictions                                            #
#--------------------------------------------------------------#

plt.rcParams['figure.figsize'] = [20, 20]

fig, ax = plt.subplots(3,2)

X = Random_df[['yearly_ConvertedComp','WorkWeekHrs']]
Y = Random_df['JobSatInt']

# Plotting Model testing Job Satisfaction predictions against Compensation
ax[0,0].scatter(JobSat_test, Xtest[:,0])
ax[0,0].set_xlabel('JobSatInd')
ax[0,0].set_ylabel('Compensation')
ax[0,0].set_xlim(3,5)
ax[0,0].set_title('Model 1(test): JobSat vs Comp')

# Plotting Model testing Job Satisfaction predictions against Work Week Hours
ax[0,1].scatter(JobSat_test, Xtest[:,1])
ax[0,1].set_xlabel('JobSatInd')
ax[0,1].set_ylabel('WorkWeekHours')
ax[0,1].set_xlim(3,5)
ax[0,1].set_ylim(0,175)
ax[0,1].set_title('Model 1(test): JobSat vs WrkWkHrs')

# Plotting Model 2 testing Job Satisfaction predictions against Compensation
ax[1,0].scatter(JobSat_test2, Xtest[:,0])
ax[1,0].set_xlabel('JobSatInd')
ax[1,0].set_ylabel('Compensation')
ax[1,0].set_xlim(3,5)
ax[1,0].set_title('Model 2(train): JobSat vs Comp')

# Plotting Model 2 testing Job Satisfaction predictions against Work Week Hours
ax[1,1].scatter(JobSat_test2, Xtest[:,1])
ax[1,1].set_xlabel('JobSatInd')
ax[1,1].set_ylabel('WorkWeekHours')
ax[1,1].set_xlim(3,5)
ax[1,1].set_ylim(0,175)
ax[1,1].set_title('Model 2(train): JobSat vs WrkWkHrs')
```

```
# Plotting Model 3 training Comensation predictions against Work Week Hours
ax[2,0].scatter(JobSat_test3, Xtest2[:,0])
ax[2,0].set_xlabel('Compensation')
ax[2,0].set_ylabel('WorkWeekHours')
ax[2,0].set_title('Model 3(train): Comp vs WrkWkHrs')

# Plotting Model 3 training Compensation predictions against NumLanguages
ax[2,1].scatter(JobSat_test3, Xtest2[:,1])
ax[2,1].set_xlabel('Compensation')
ax[2,1].set_ylabel('NumLanguages')
ax[2,1].set_title('Model 3(train): Comp vs NumLanguages')
```

```
[50]: Text(0.5, 1.0, 'Model 3(train): Comp vs NumLanguages')
```



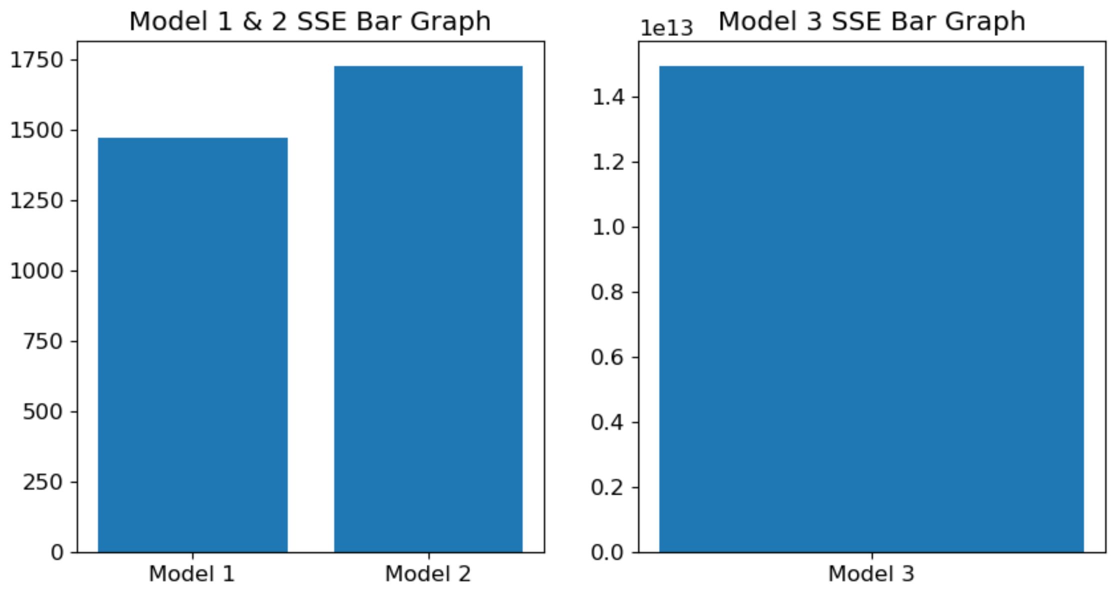
4.0.17 Graph Explanations

- The first graph pictured above on the far top left is our model 1 prediction of the job satisfaction index against yearly compensation from the dataset
- The second graph pictured above on the far top right is our model 1 prediction of the job satisfaction index against work week hours from the dataset
- The third graph pictured above in the center left position is our model 2 prediction of the job satisfaction index against yearly compensation from the dataset
- The fourth graph pictured above in the center right position is our model 2 prediction of the job satisfaction index against work week hours from the dataset
- The fifth graph pictured above on the far bottom left is our model 3 prediction of yearly compensation against work week hours from the dataset
- The sixth graph pictured above on the far bottom right is our model 3 prediction of yearly compensation against number of languages known from the dataset

4.0.18 Caculations and Plotting of SSE (Models 1, 2, & 3)

[51]:
```
#--------------------------------------------------------------#
# This cell is dealing with the calculations and the plotting #
# of the SSE's of models 1, 2 and 3                            #
#--------------------------------------------------------------#

plt.rcParams['figure.figsize'] = [10, 5]

fig, ax = plt.subplots(1,2)

SSE_M1 = 0
counter = 0
for i in JobSat_test:
    SSE_M1 += (i - Ytest[counter])**2
    counter += 1

SSE_M2 = 0
counter = 0
for i in JobSat_test2:
    SSE_M2 += (i - Ytest[counter])**2
    counter += 1

SSE_M3 = 0
counter = 0
for i in JobSat_test3:
    SSE_M3 += (i - Ytest2[counter])**2
    counter += 1


labels = ['Model 1', 'Model 2']
data = [SSE_M1, SSE_M2]

label = ['Model 3']
data1 = [SSE_M3]

ax[0].bar(labels, data)
ax[1].bar(label, data1)
ax[0].set_title("Model 1 & 2 SSE Bar Graph")
ax[1].set_title("Model 3 SSE Bar Graph")
```

[51]:
```
Text(0.5, 1.0, 'Model 3 SSE Bar Graph')
```



4.0.19 Graph Explanations

- The first graph pictured above on the left is the total sum squared error from both models 1 and 2 represented in a bar format
- The second graph pictured above on the right is the total sum squared error from just model 3

[ ]: