Sucheta Naik

182000492

1. Implementing kernel ridge regression and compare to regular ridge regression.

This code will import libraries

In [35]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

This code will load data in csv format

In [36]:
```
dataset = pd.read_csv('/Users/suchetanaik/Desktop/Statistics & Machine Learning/USA_Housing.csv')
#dataset.head()
```

Out[36]:

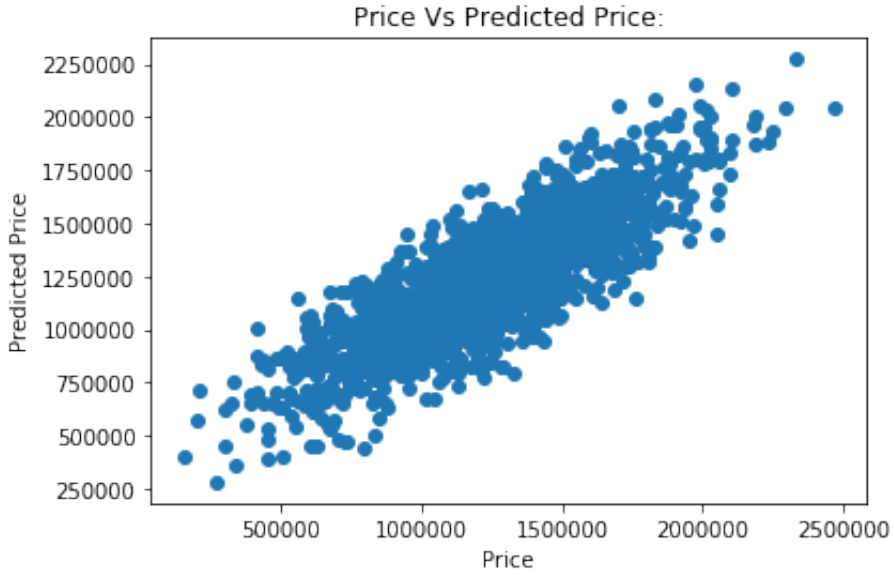
	Avg. Area Income	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.45857	7.009188	4.09	23086.80050	1.059034e+06
1	79248.64245	6.730821	3.09	40173.07217	1.505891e+06
2	61287.06718	8.512727	5.13	36882.15940	1.058988e+06
3	63345.24005	5.586729	3.26	34310.24283	1.260617e+06
4	59982.19723	7.839388	4.23	26354.10947	6.309435e+05

In [55]:
```
#Independent Variables
X = dataset.drop('Price', axis=1)

#target variable
y = dataset.iloc[:,4].values
```

In [76]:
```
from sklearn.model_selection  import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
=0.33, random_state=42)
```

In [85]:
```
from sklearn.linear_model import Ridge
lr = Ridge()
lr.fit(X_train, y_train)
Y_pred = lr.predict(X_test)
plt.scatter(y_test, Y_pred)
plt.xlabel("Price")
plt.ylabel("Predicted Price")
plt.title("Price Vs Predicted Price:")
plt.show()
```



This code will evaluate using R Square

In [75]:
```
lr.score(X_test,y_test)
```

Out[75]: 0.69398351562559091

In [91]:
```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y
_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_p
red))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_erro
r(y_test, y_pred)))
```

```
Mean Absolute Error: 195342.060336
Mean Squared Error: 60209662690.3
Root Mean Squared Error: 245376.57323
```

In [92]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = sklearn.model_selection.train_te
st_split(X , y_opt, test_size = 0.33, random_state = 5)
```

In [93]:
```
from sklearn.kernel_ridge import KernelRidge
```

In [94]:
```
lm = KernelRidge()
lm.fit(X_train, y_train)
y_pred = lm.predict(X_test)
plt.scatter(y_test, y_pred)
plt.xlabel("Price")
plt.ylabel("Predicted Price")
plt.title("Price vs predicted Price:")
plt.show()
```

This code will evaluate using R Square

In [104]:
```
lm.score(X_test,y_test)
```

Out[104]: 0.50216022235590863

Since R-squared is a statistical measure of how close the data are to the fitted regression line, we can say that Regular Ridge Regression is the best regression model for USA_Housing dataset (R squared value = 0.69)

2.Comparison between linear, polynomial, and RBF kernels in SVM

Support Vector Machine (SVM)

Age and Salary are independent variables to predict if that particular person will click on the social media Advertisement or not.

This code will load the dataset

In [2]:
```
dataset = pd.read_csv('/Users/suchetanaik/Desktop/Statistics & Machine Learning/PaymentTransaction.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

In [3]:
```
dataset.head(10)
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

This code will split the dataset into the Training set and Test set

In [27]:
```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

In [29]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

This code will fit SVM to the Training set

In [ ]:
```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

This code will predict the test set results

In [7]:
```
y_pred = classifier.predict(X_test)
```

This code will import confusion matrix

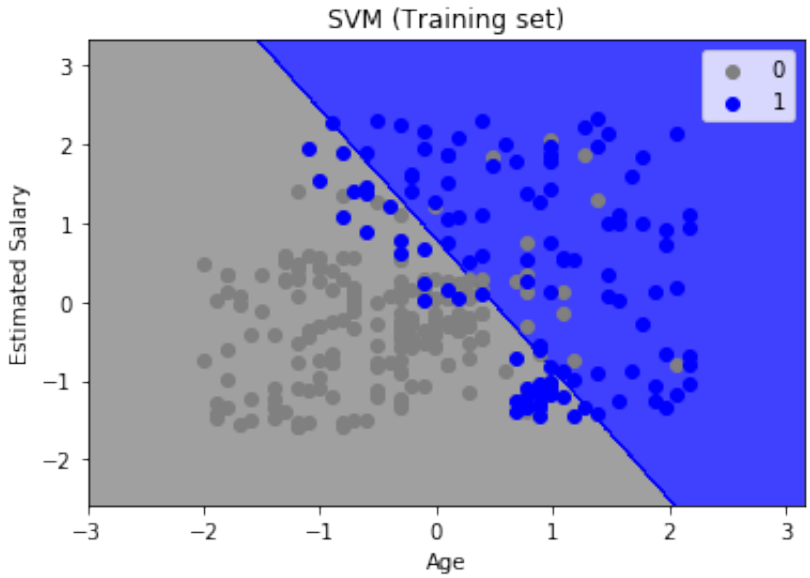
In [8]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [9]:
```
cm
```

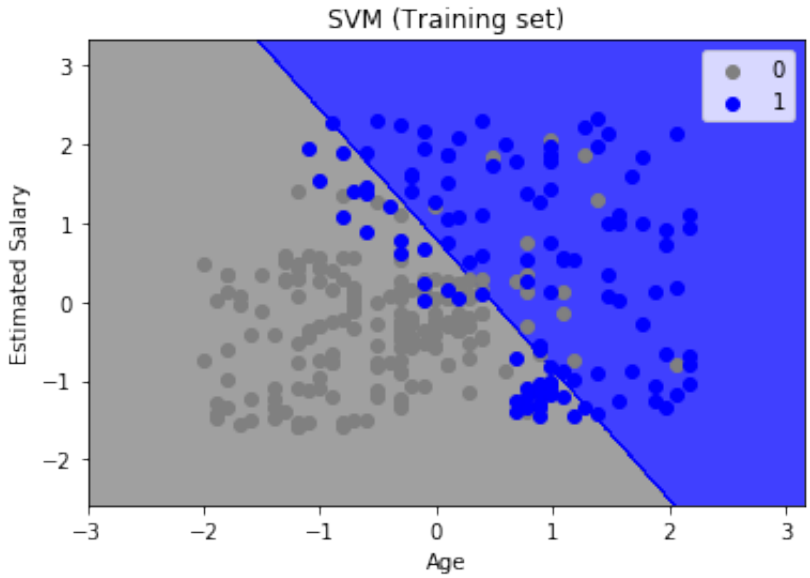
Out[9]:
```
array([[66,  2],
       [ 8, 24]], dtype=int64)
```

Incorrect Predictions = 10

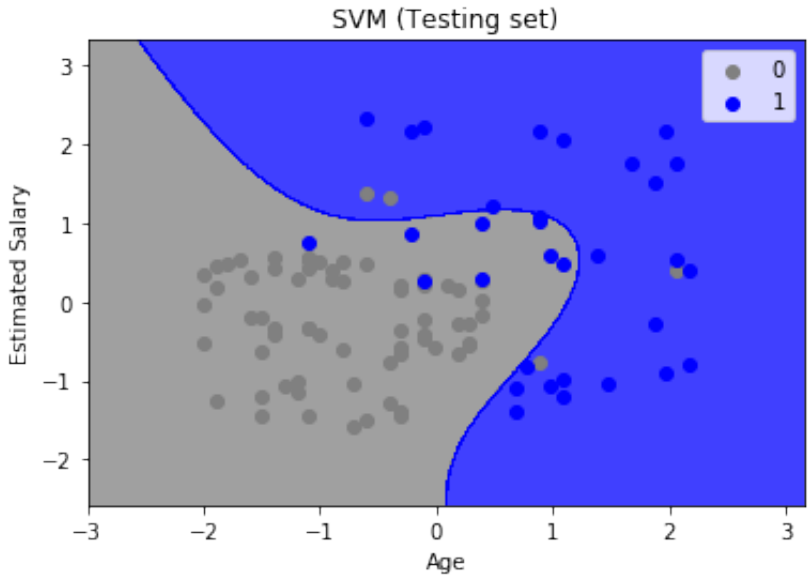
In [10]:
```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
vel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('grey', 'blue'))(i), label = j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



In [11]:
```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
vel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('grey', 'blue'))(i), label = j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



In [31]:
```
# Visualising the Testing set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
vel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('grey', 'blue'))(i), label = j)
plt.title('SVM (Testing set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



This code will fit svm to training set

In [13]:
```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[13]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='r
bf',
  max_iter=-1, probability=False, random_state=0, shrinking=True,
  tol=0.001, verbose=False)
```

This code will predict the test set results

In [14]:
```
y_pred = classifier.predict(X_test)
```

This code will import the confusion matrix

In [15]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

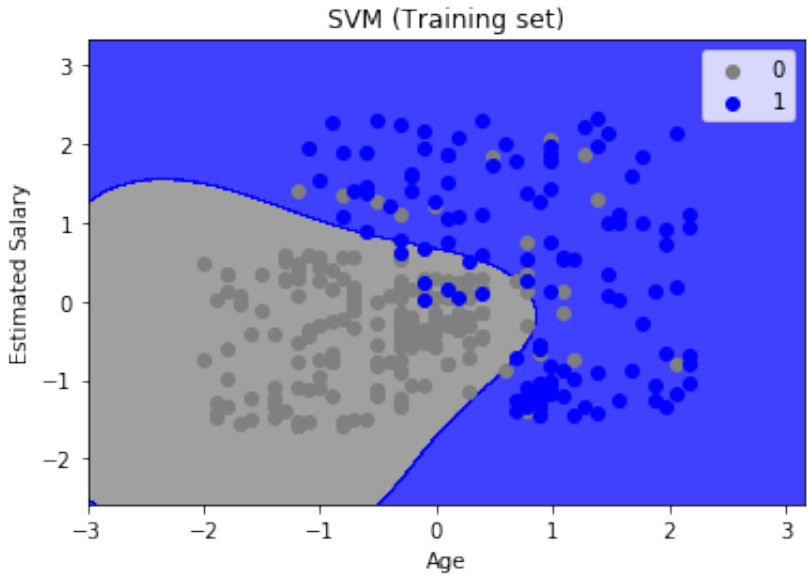
In [16]:
```
cm
```

Out[16]:
```
array([[64,  4],
       [ 3, 29]], dtype=int64)
```

Incorrect Predictions = 7

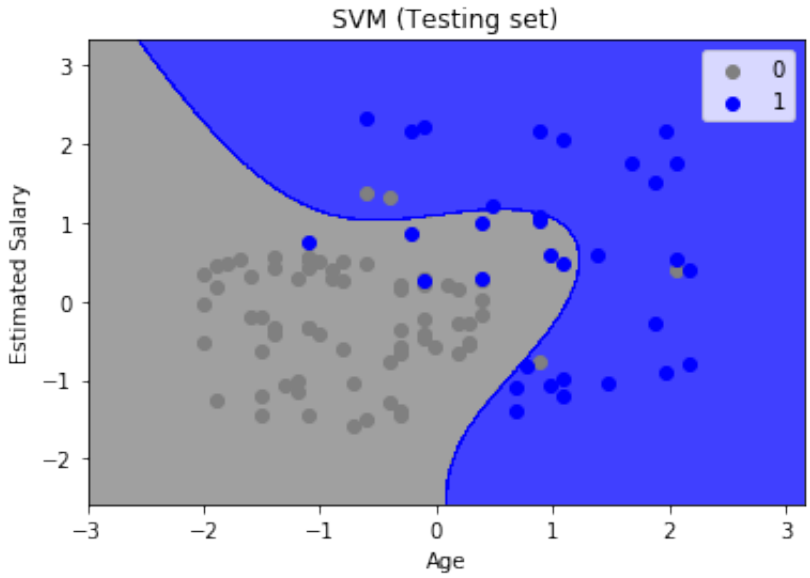
This code will visualize the Training set results

In [17]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
vel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('grey', 'blue'))(i), label = j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



This code will visualize the Testing set results

In [33]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
vel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('grey', 'blue'))(i), label = j)
plt.title('SVM (Testing set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



This code will fit SVM to Training set

In [19]:
```
# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'poly', degree = 3, random_state = 0, coe
f0=0.1)
classifier.fit(X_train, y_train)
```

Out[19]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.1,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='p
oly',
  max_iter=-1, probability=False, random_state=0, shrinking=True,
  tol=0.001, verbose=False)
```

This code will predict the test set results

In [20]:
```
y_pred = classifier.predict(X_test)
```

This code will import confusion matrix

In [21]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

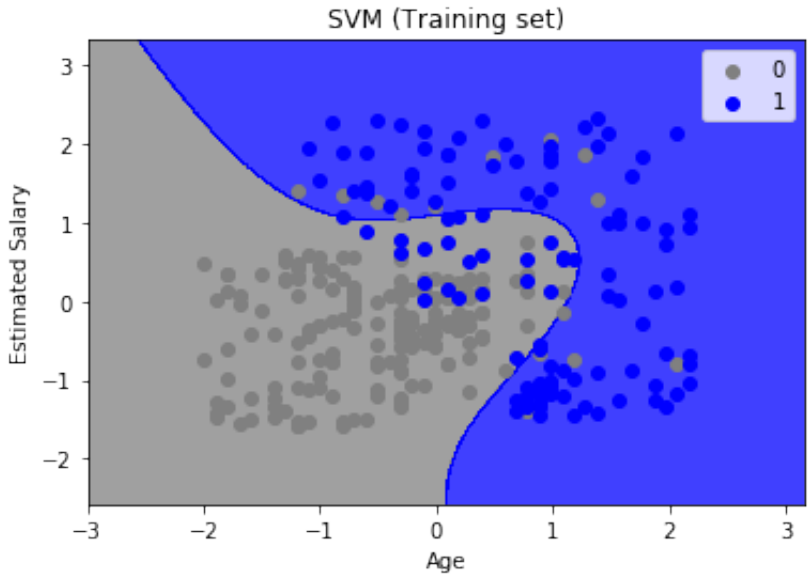
In [22]:
```
cm
```

Out[22]:
```
array([[64,  4],
       [ 9, 23]], dtype=int64)
```

Incorrect Predictions = 13

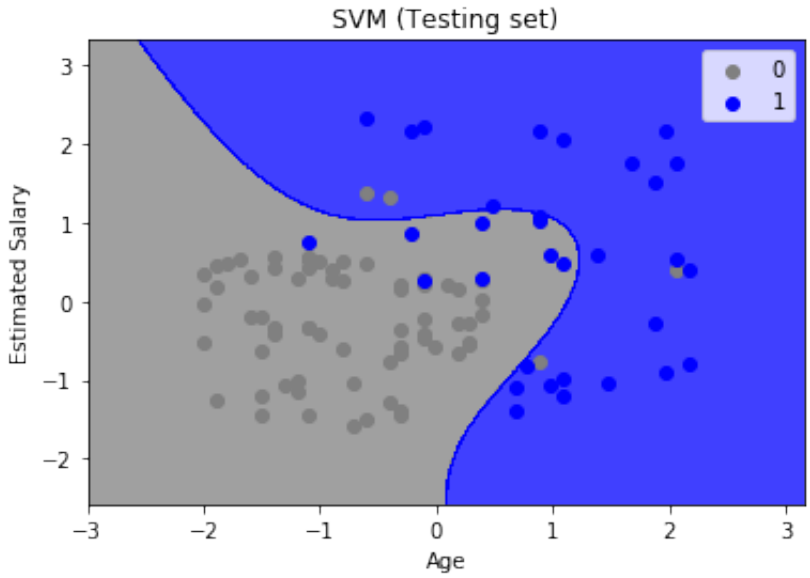
This code will visualize the Training set results

In [25]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
vel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('grey', 'blue'))(i), label = j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



This code will visualize the Testing set results

In [32]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
vel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('grey', 'blue'))(i), label = j)
plt.title('SVM (Testing set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



We can say RBF performs better on this dataset compared to linear,polynomial kernel.